# JBoss Operations Network 3.1
# Managing Resource Configuration

for editing resource settings and configuration through the JBoss ON UI
Edition 3.1.1

Landmann

# JBoss Operations Network 3.1 Managing Resource Configuration

for editing resource settings and configuration through the JBoss ON UI
Edition 3.1.1

Landmann
rlandmann@redhat.com

**Abstract**

JBoss Operations Network can view and edit configuration files and settings for many types of managed resources. This allows administrators to review, audit, and initiate changes across multiple platforms and resources consistently. This guide provides GUI-based procedures to manage resource configuration.

# Table of Contents

## 1. Document Information

This guide is part of the overall set of guides for users and administrators of JBoss ON. Our goal is clarity, completeness, and ease of use.

### 1.1. Giving Feedback

If there is any error in this guide or there is any way to improve the documentation, please let us know. Bugs can be filed against the documentation for the community-based RHQ Project in Bugzilla, http://bugzilla.redhat.com/bugzilla. Make the bug report as specific as possible, so we can be more effective in correcting any issues:

1. Select the **`Other`** products group.
2. Select **`RHQ Project`** from the list.
3. Set the component to **`Documentation`**.
4. Set the version number to 3.1.
5. For errors, give the page number (for the PDF) or URL (for the HTML), and give a succinct description of the problem, such as incorrect procedure or typo.

   For enhancements, put in what information needs to be added and why.
6. Give a clear title for the bug. For example, **`"Incorrect command example for setup script options"`** is better than **`"Bad example"`**.

We appreciate receiving any feedback — requests for new sections, corrections, improvements, enhancements, even new ways of delivering the documentation or new styles of docs. You are welcome to contact Red Hat Content Services directly at docs@redhat.com.

### 1.2. Document History

| | | |
|---|---|---|
| **Revision 3.1.1-0.400** | **2013-10-31** | **Rüdiger Landmann** |
| Rebuild with publican 4.0.0 | | |
| **Revision 3.1.1-0** | **October 3, 2012** | **Ella Deon Lackey** |
| Updates and bug fixing for JBoss Operations Network 3.1.1. | | |
| **Revision 3.1-0** | **June 12, 2012** | **Ella Deon Lackey** |
| Initial release of JBoss Operations Network 3.1. | | |

## 2. Summary: Using JBoss ON to Make Changes in Resource Configuration

One of the most basic parts of managing your applications, servers, and services is the simple ability to change their configuration.

JBoss Operations Network allows you to view the current configuration for many resource types directly in the JBoss ON UI, without having to access the platform's filesystem directly. Even more, JBoss ON allows you to edit the configuration directly for a single resource or for an entire group of compatible resources.

JBoss ON has three key ways that administrators can manage resource configuration:

- *Directly edit resource configuration.* JBoss ON can edit the configuration files of a variety of different managed resources through the JBoss ON UI.
- *Audit and revert resource configuration changes.* For the specific configuration files that JBoss ON manages for supported resources, you can view individual changes to the configuration properties and revert them to any previous version.
- *Define and monitor configuration drift.* System configuration is a much more holistic entity than specific configuration properties in specific configuration files. Multiple files for an application or even an entire platform work together to create an optimum configuration. *Drift* is the (natural and inevitable) deviation from that optimal configuration. Drift management allows you to define what the baseline, desired configuration is and then tracks all changes from that baseline.

This section has a very general overview of these three ways of managing resource configuration. More detailed descriptions and procedures are in the subsequent sections.

## 2.1. Easy, Structured Configuration

Basic configuration files use simple key-value pairs to define information.

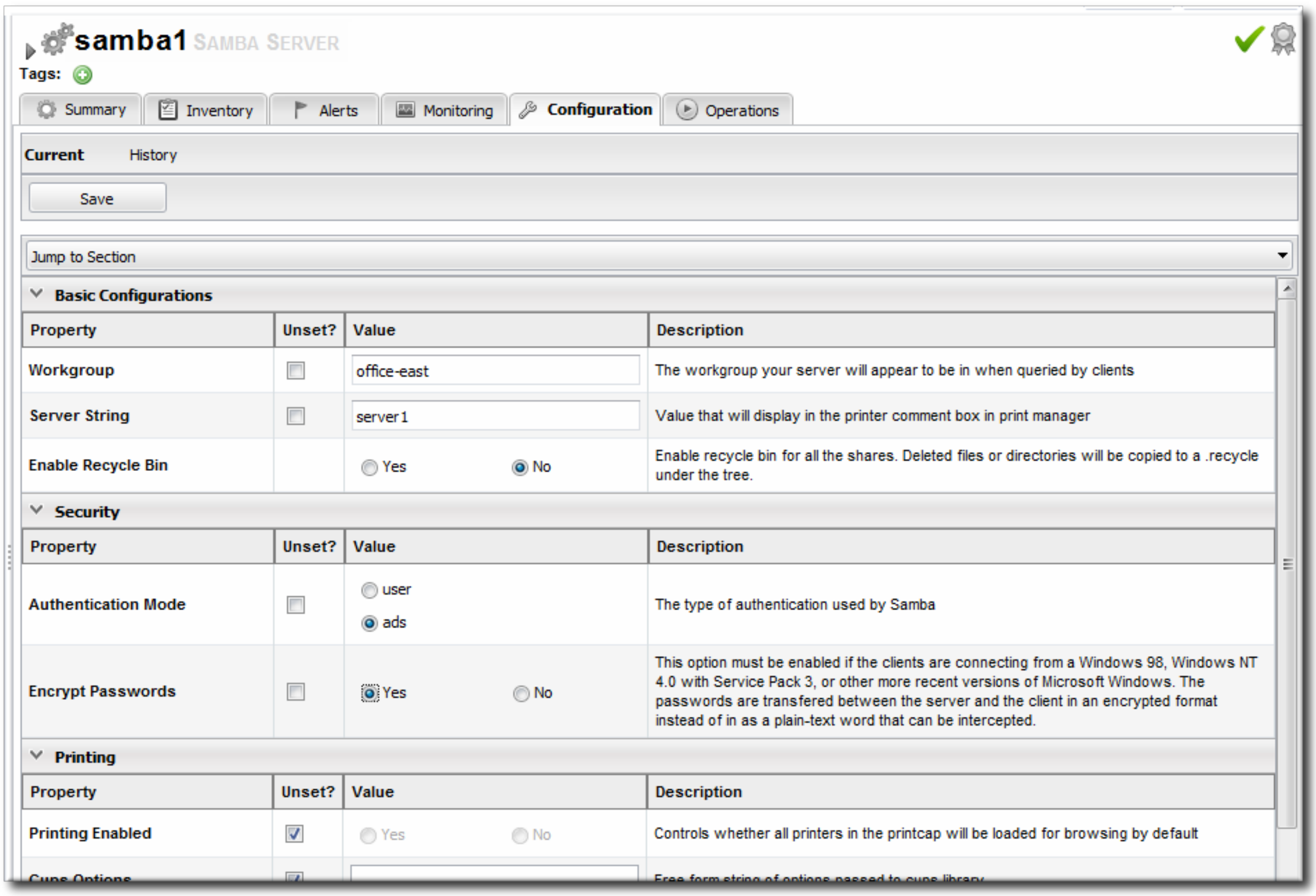
```
key1 = value1
key2 value2
```

These are *simple properties*, representing strings, numbers, or booleans — any type of information where there is one value per key.

JBoss ON also supports resource configuration using *complex properties*, which may be a list of values or a map of values (essentially a table of lists).

```
<default-configuration>
    <ci:list-property name="my-list">
        <c:simple-property name="element" type="string"/>
        <ci:values>
           <ci:simple-value value="a"/>
           <ci:simple-value value="b"/>
           <ci:simple-value value="c"/>
        </ci:values>
    </ci:list-property>
</default-configuration>
```

JBoss ON parses the configuration files — both simple properties and complex properties — and then renders a structured, easy-to-follow form in the JBoss ON GUI. Simple properties are displayed with fields or radio buttons as appropriate, while complex properties are displayed with menus or other selection options.



samba1 SAMBA SERVER

Tags:

Summary | Inventory | Alerts | Monitoring | **Configuration** | Operations

**Current** History

Save

Jump to Section

**Basic Configurations**

| Property | Unset? | Value | Description |
|---|---|---|---|
| Workgroup | ☐ | office-east | The workgroup your server will appear to be in when queried by clients |
| Server String | ☐ | server1 | Value that will display in the printer comment box in print manager |
| Enable Recycle Bin | | Yes [✓ No] | Enable recycle bin for all the shares. Deleted files or directories will be copied to a .recycle under the tree. |

**Security**

| Property | Unset? | Value | Description |
|---|---|---|---|
| Authentication Mode | ☐ | user [✓ ads] | The type of authentication used by Samba |
| Encrypt Passwords | ☐ | [✓ Yes] No | This option must be enabled if the clients are connecting from a Windows 98, Windows NT 4.0 with Service Pack 3, or other more recent versions of Microsoft Windows. The passwords are transfered between the server and the client in an encrypted format instead of in as a plain-text word that can be intercepted. |

**Printing**

| Property | Unset? | Value | Description |
|---|---|---|---|
| Printing Enabled | ✓ | Yes No | Controls whether all printers in the printcap will be loaded for browsing by default |
| Cups Options | ✓ | | Free form string of options passed to cups library |

**Figure 1. Configuration Form for a Samba Server**

The structured configuration form makes it easy for you to view the current configuration quickly.

The structured form also makes it possible for JBoss ON to validate that the configuration properties have valid formats before saving changes.

**NOTE**

JBoss ON only validates that the given value matches the required format for that property. It does not validate that the value given is reasonable or allowed for that resource property.

Performing configuration changes in JBoss ON has major benefits for IT administrators:

- There is instant validation on the format of properties that are set through the UI.
- Audit trails for all configuration changes can be viewed in the resource history for both external and JBoss ON-initiated configuration changes.
- Configuration changes can be reverted to a previous stable state if an error occurs.
- Configuration changes can be made to groups of resource of the same type, so multiple resources (even on different machines) can be changed simultaneously.
- Alerts can be used in conjunction with configuration changes, either simply to send automatic announcements of any configuration changes or to initiate operations or scripts on related resources as configuration changes are made.
- Access control rules are in effect for configuration changes, so JBoss ON users can be prevented

from viewing or initiating changes on certain resources.

## 2.2. Identifying What Configuration Properties Can Be Changed

JBoss ON supports configuration change for an extensive array of resources — including hosts and sudoers files, Samba servers, Postfix servers, databases, web app contexts, cron tabs, web servers, and scripts.

Any resource which supports configuration changes through JBoss ON has a **`Configuration`** tab on its resource page.

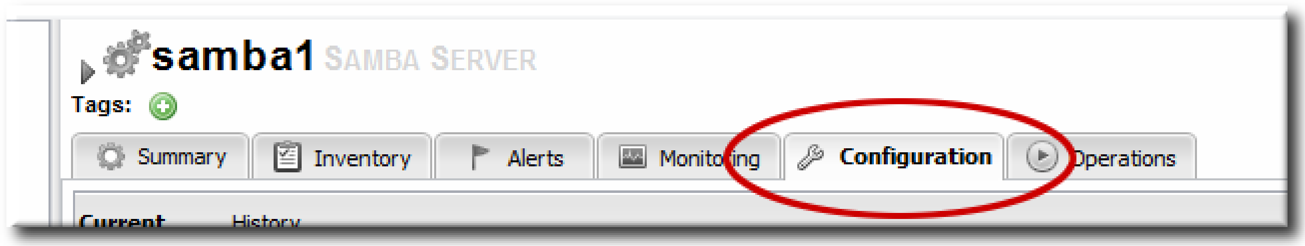


**Figure 2. Configuration Tab**

For a complete reference of the configuration properties for each resource, see the *Resource Reference: Monitoring, Operation, and Configuration Options*.

## 2.3. Auditing and Reverting Resource Configuration Changes

Tracking for configuration changes is a crucial part of systems administration. It's important for maintenance, for performance, and for incident recovery — particularly when it is possible to revert change or correlate changes to incidents.

Every time a change is made to the resource configuration, whether through JBoss ON GUI or on the resource itself, the change is detected by JBoss ON and logged with a revision number. When a change is made outside of JBoss ON, the change is simply noted. When the change is made through the JBoss ON UI, the timestamp and the name of the user who made the change are both recorded.

Every change is recorded in a history, and the different changesets can be viewed and compared to one another. One change can be selected and the resource configuration can be rolled back to that selected change.

Tracking the configuration history and reverting changes is covered more in Section 4.1, “Tracking and Comparing Configuration Changes” and Section 4.2, “Reverting Configuration Changes”.

## 2.4. Tracking Configuration Drift

Much of the JBoss ON configuration management is designed around *implementing* changes for resources by editing configuration files or updating files and packages. But another aspect of managing configuration is *detecting changes*.

IT administrators must invest a significant amount of time planning the optimum configuration for systems in every type of environment, from production systems to internal resources. This ideal configuration includes file settings, software versions, and system settings. Resource configuration is going to change naturally over time, but administrators need to be able to track those changes to make sure that no unplanned or undesirable changes impact the resource. Defining a baseline configuration and tracking changes helps systems remain resilient during both maintenance and failures.

The unplanned changes that occur to a resource's configuration is called *drift*, as the configuration moves away from the designed baseline. Drift is common because of frequent software and hardware updates, particularly in a colocation facility or using virtual machines.

Production, staging, development, and recovery configurations are designed to have identical or near-identical configuration to maintain consistency. As the configurations within the different environments change, there emerges a *configuration gap*. Ultimately, this configuration gap can lead to disaster recovery failures or high availability failures because the configuration of the production system and the backup system are too different.

*Drift monitoring* provides a very general, freewheeling *content monitoring*. Rather than structured configuration management, drift monitoring tracks changes, any changes, in files — even binary files.

### Configuration History v. Configuration Drift

The configuration history for a resource applies only to the supported configuration properties for that specific resource instance.
Drift management has a much more external view of configuration changes. Drift is associated with a resource — like a platform or a JBoss server — but it is not restricted to that resource or to set properties for that resource:

- Drift looks at whole files within a directory, including added and deleted files and binary files.
- Drift supports user-defined templates which can be applied to any resource which supports drift monitoring.
- Drift can keep a running history of changes where each changeset (*snapshot*) is compared against the previous set of changes. Alternatively, JBoss ON can compare each change against a defined baseline snapshot.

The drift definition that is essentially a profile that identifies a directory and files that should be monitored. Any time there is any change in that drift base directory or any of its subdirectories — file modifications, new files, or deleted files — the drift detection scan notices the change and records it.

Drift detection can be used by administrators to track scheduled changes, maintenance and updates, and server changes. There are a lot of common scenarios where administrators need to be aware that change has occurred (and even be able to identify the specific changes made), but that occur in areas outside normal JBoss ON configuration tracking:

- System password changes
- System ACL changes
- Database and server URL changes
- JBoss settings changes
- Changed JAR, WAR, and other binary files used by applications
- Script changes

**TIP**

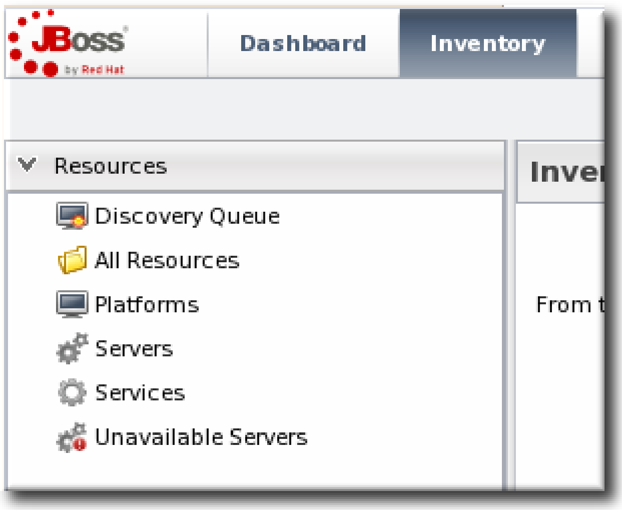
Drift is not bound or restricted to a resource managed by JBoss ON. You can create a drift definition for a platform and set it to monitor any file or directory on that platform, even if it is outside the JBoss ON inventory, as long as that directory is accessible to the system user that the JBoss ON agent runs as.

Managing configuration drift is described more in Section 5, “Managing Configuration Drift”.

## 3. Changing the Configuration for a Resource

### 3.1. Changing the Configuration on a Single Resource

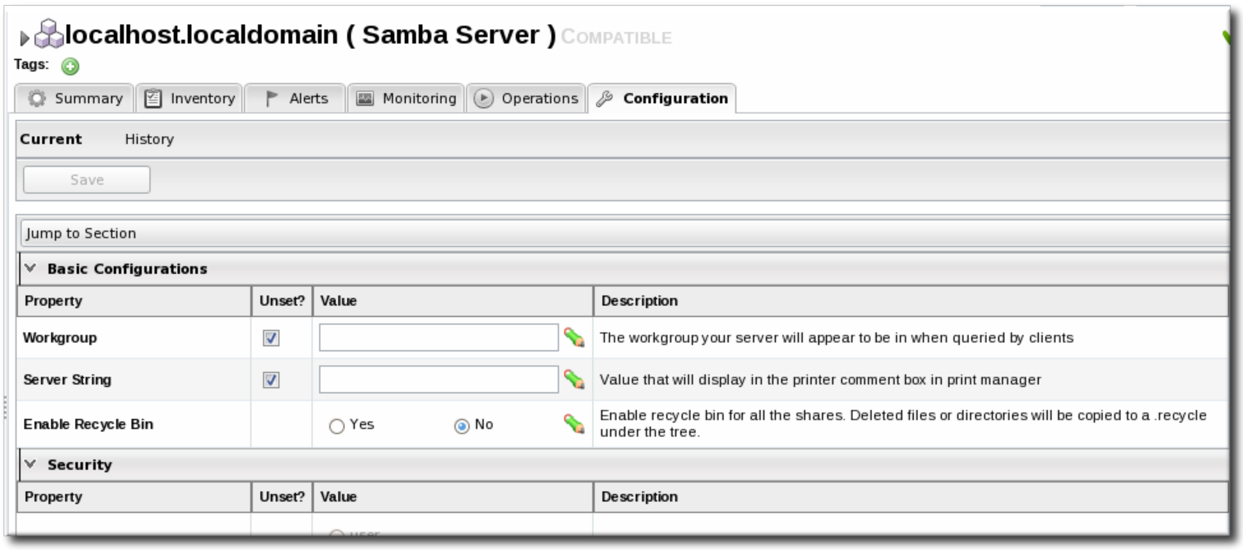
1. Click the **`Inventory`** tab in the top menu.
2. Select the resource type in the **`Resources`** menu table on the left, and then browse or search for the resource.



3. Open the **`Configuration`** tab for the resource.
4. Click the **`Current`** subtab.
5. To edit a field, make sure the **`Unset`** checkbox is *not* selected. The **`Unset`** checkbox means that JBoss ON won't submit any values for that resource and any values are taken from the resource itself.

   Then, make any changes to the configuration.

   The list of available configuration properties, and their descriptions, are listed for each resource type in the *Resource Reference: Monitoring, Operation, and Configuration Options*.



Basic Configurations

| Property | Unset? | Value | Description |
|---|---|---|---|
| Workgroup | ✓ | | The workgroup your server will appear to be in when queried by clients |
| Server String | ✓ | | Value that will display in the printer comment box in print manager |
| Enable Recycle Bin | | Yes [✓ No] | Enable recycle bin for all the shares. Deleted files or directories will be copied to a .recycle under the tree. |

Security

| Property | Unset? | Value | Description |
|---|---|---|---|

6. Click the **Save** button at the top of the properties list.

## 3.2. Changing the Configuration for a Compatible Group

Similar to other templating functions in JBoss ON, like alert templates, configuration changes can be made on a compatible or autogroup, so that all of the members of that group can be up updated simultaneously with the same settings.

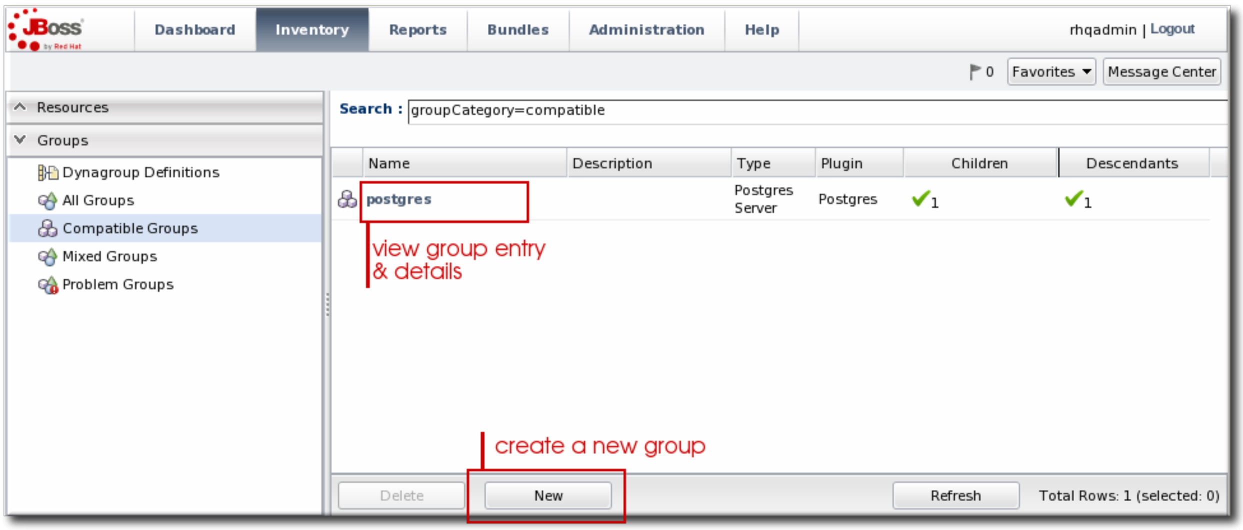
**NOTE**

To change the current configuration for a group, a few conditions must be true so that the current group configuration can be reliably calculated for the individual resource configurations:

- The group members must all be the same resource type.
- All group member resources must be available (**UP**).
- No other configuration update requests can be in progress for the group or any of its member resources.
- The current member configurations must be successfully retrieved from the agents.

The process for setting the configuration for a group is the same as setting it for an individual resource:

1. Click the **Inventory** tab in the top menu.
2. In the **Groups** box in the left menu, select the **Compatible Group** link.



Search : groupCategory=compatible

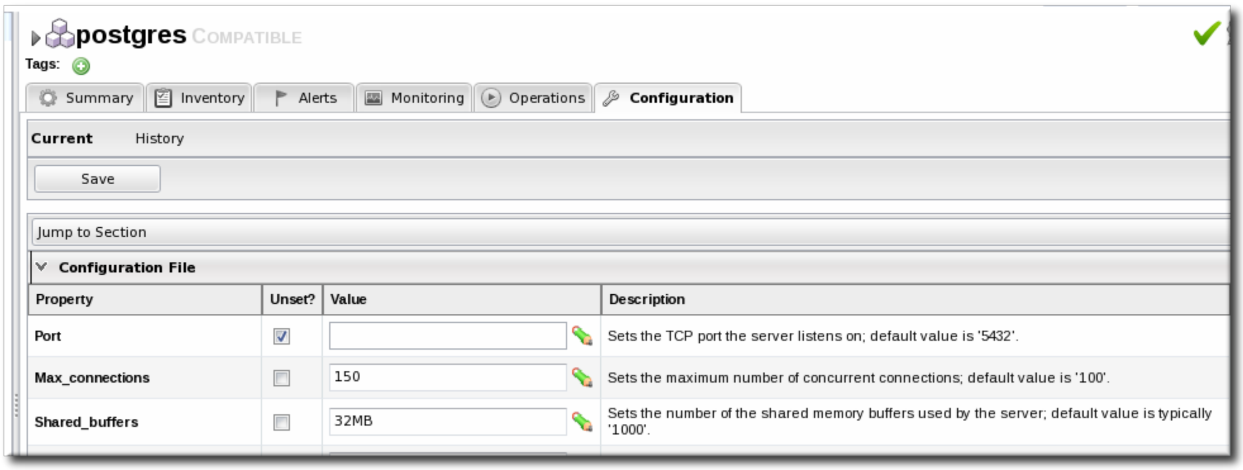
| Name | Description | Type | Plugin | Children | Descendants |
|---|---|---|---|---|---|
| postgres | | Postgres Server | Postgres | ✓1 | ✓1 |

Total Rows: 1 (selected: 0)

3. Select the group to edit.
4. Open the **`Configuration`** tab.
5. Click the **`Current`** subtab.
6. To edit a field, make sure the **`Unset`** checkbox is *not* selected. The **`Unset`** checkbox means that JBoss ON won't submit any values for that resource and any values are taken from the resource itself.

   Then, make any changes to the configuration.

   The list of available configuration properties, and their descriptions, are listed for each resource type in the *Resource Reference: Monitoring, Operation, and Configuration Options*.



| Property | Unset? | Value | Description |
|---|---|---|---|
| Port | ✓ | | Sets the TCP port the server listens on; default value is '5432'. |
| Max_connections | ☐ | 150 | Sets the maximum number of concurrent connections; default value is '100'. |
| Shared_buffers | ☐ | 32MB | Sets the number of the shared memory buffers used by the server; default value is typically '1000'. |

**TIP**

It is possible to change the configuration for all members by editing the form directly, but it is also possible to change the configuration for a subset of the group members. Click the green pencil icon, and then change the configuration settings for the members individually.

7. Click the **`Save`** button at the top of the form.

## 3.3. Editing Script Environment Variables

Scripts are autodetected on a server, as are other applications and services on the machine. Scripts can be configured and managed like any other resource, which means that JBoss ON allows you to both define configuration settings for and set up operations to run the scripts in inventory.

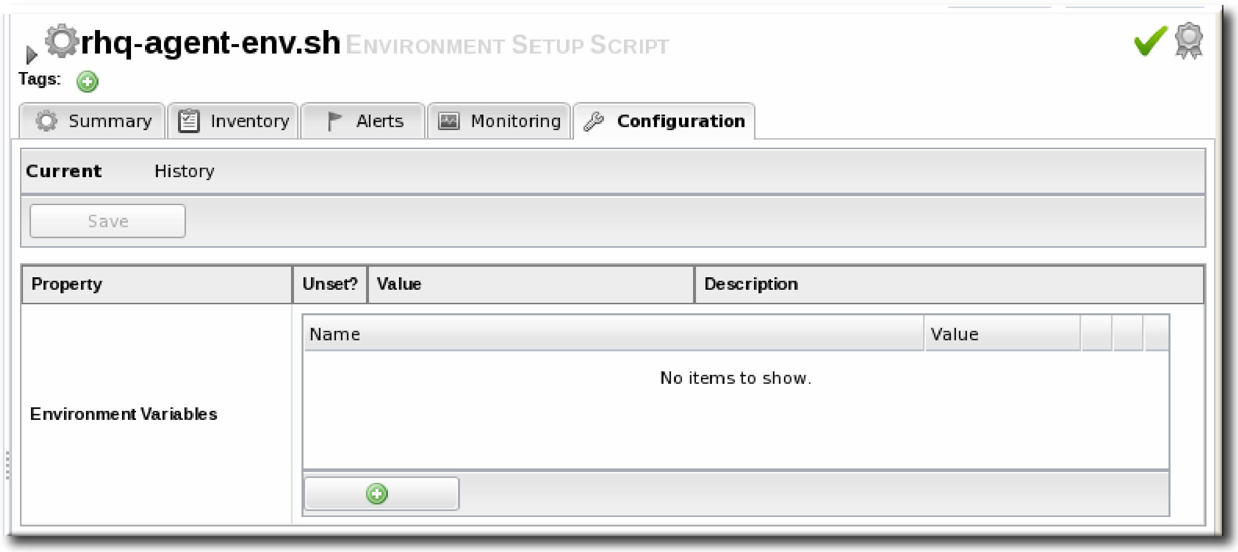
Whether a script is added or detected, there are only two configuration areas for the inventory entry: the path to the script, which places the script within the hierarchy, and any environment variables that should be set with the script.

These environment variables can be added and edited even after the script is imported:

**IMPORTANT**

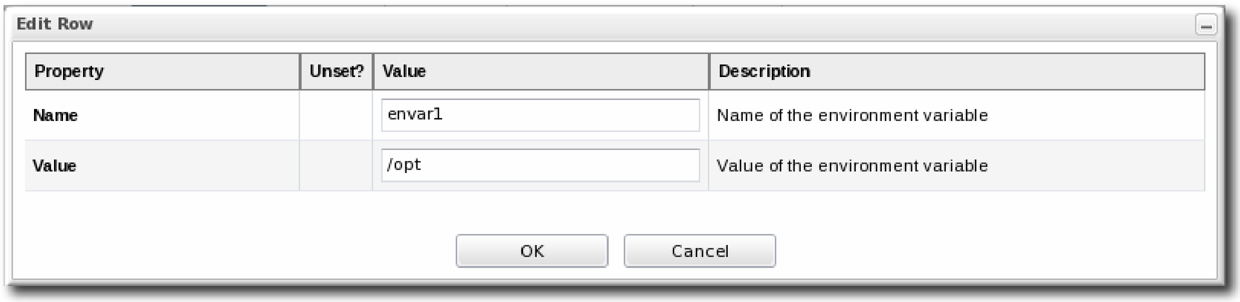
Before setting environment variables in the JBoss ON configuration, make sure that the environment *on the resource* is already configured properly.

1. Click the **`Inventory`** tab in the top menu.
2. Search for the script resource.
3. Open the **`Configuration`** tab for the script resource.
4. Click the green plus sign (+) to add an environment variable.



| Property | Unset? | Value | Description |
|---|---|---|---|
| Environment Variables | | Name / Value: No items to show. | |

5. Enter the environment variable. Each new environment variable has the format *name=value;* and is added on a new line.



| Property | Unset? | Value | Description |
|---|---|---|---|
| Name | | envar1 | Name of the environment variable |
| Value | | /opt | Value of the environment variable |

If the variable's value contains properties with the syntax **`%propertyName%`**, then JBoss ON interprets the value as the current values of the corresponding properties from the script's parent resource's connection properties.

6. After resetting an environment variable, restart the JBoss ON agent to propagate the changes. If the agent isn't restarted, new variables will not be propagated to the resource and will not resolve when the script is next executed, even if the configuration is correct.

**TIP**

Add the line **`@echo off`** in Windows scripts to prevent echoing the executed commands along with the execution results.

## 3.4. Configuring Apache for Configuration Management (Deprecated)

JBoss ON manages configuration on Apache resources using an Augeas lens. A special version of Augeas is included with the JBoss ON agent which enables Apache configuration management. That Augeas lens must be enabled on the Apache resource for configuration management to work.

### 3.4.1. Considerations and Notes for Apache Configuration Management

#### Deprecated Augeas Plug-in

Apache configuration management is supported through a special Augeas agent plug-in, which connects with and manages the Augeas lens on the Linux instance. The Augeas agent plug-in is deprecated in JBoss ON 3.1.1 and may be removed in a later release.

#### Augeas and Apache Monitoring

**The Augeas lens is not required for Apache monitoring. It is only used for Apache configuration management.** An Apache resource can be monitored, with alerting, operations, and all other management tasks available without any additional configuration. The Augeas lens is used only for editing the Apache configuration files and virtual hosts through JBoss ON.

#### Supported Platforms for Apache Configuration

Apache configuration management is only supported for Apache instances installed on Linux.

#### Disabling noexec Options for Apache Directories

If the **`/tmp`** directory is configured a **`noexec`** in the **`fstab`** file, the agent throws exceptions because it cannot properly initialize the Augeas lens. In that case, the **`Configuration`** tab is unavailable for the Apache resource.

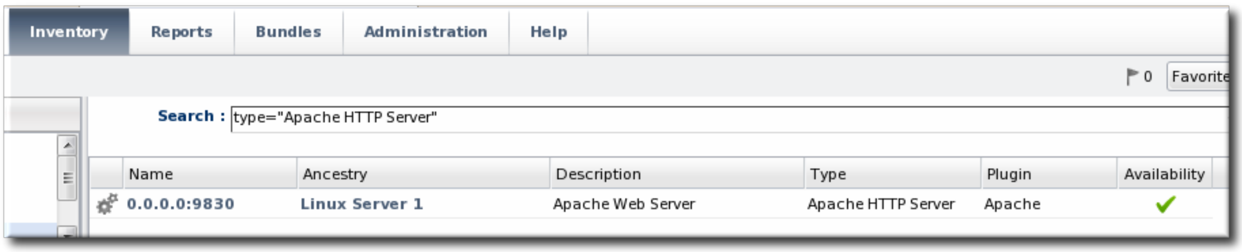
Make sure that the **`/tmp`** directory does not have **`noexec`** set as an option.

```
#
# /etc/fstab
#
tmpfs /dev/shm tmpfs defaults 0 0
devpts                  /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                   /sys                    sysfs   defaults        0 0
proc                    /proc                   proc    defaults        0 0
```

### 3.4.2. Enabling Configuration Management

Apache configuration management is configured as one of the connection settings for the Apache resource, which sets how the agent connects to the resource.

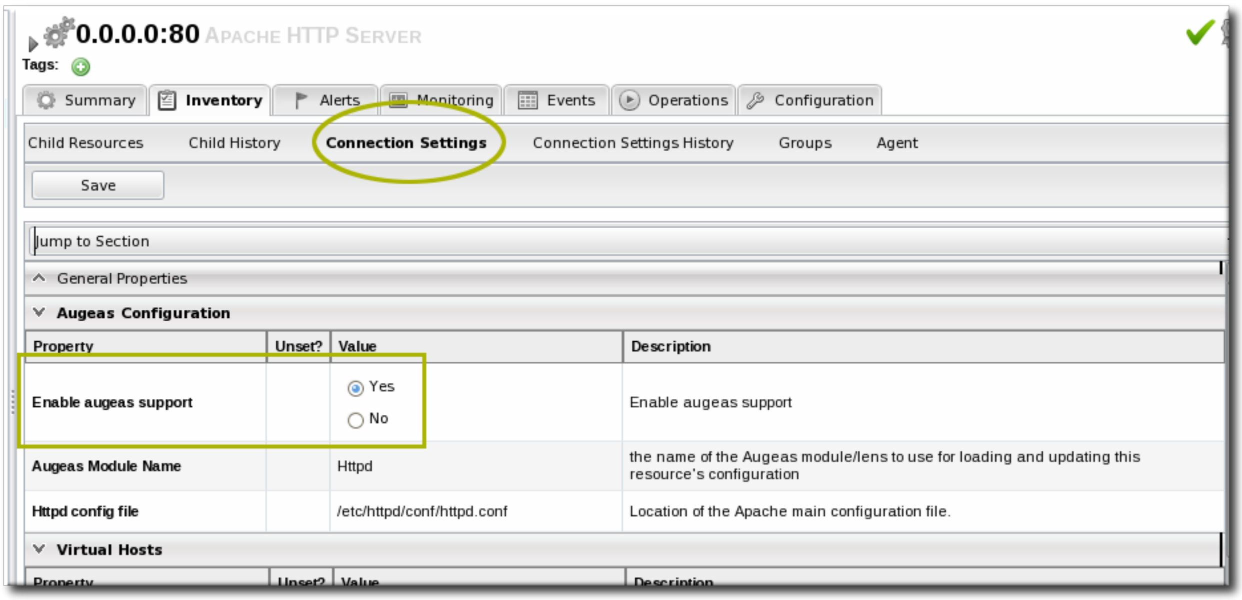
1. Click the **`Inventory`** tab in the top menu.
2. Select the resource type in the **`Resources`** menu table on the left, and then search for the Apache resource.



Search : type="Apache HTTP Server"

| Name | Ancestry | Description | Type | Plugin | Availability |
|---|---|---|---|---|---|
| 0.0.0.0:9830 | Linux Server 1 | Apache Web Server | Apache HTTP Server | Apache | ✓ |

3. Click the IP address of the Apache instance.
4. Open the **`Inventory`** tab, then click the **`Connections`** subtab.



| Property | Unset? | Value | Description |
|---|---|---|---|
| Enable augeas support | | [✓ Yes] No | Enable augeas support |
| Augeas Module Name | | Httpd | the name of the Augeas module/lens to use for loading and updating this resource's configuration |
| Httpd config file | | /etc/httpd/conf/httpd.conf | Location of the Apache main configuration file. |

5. Jump to the **`Augeas Configuration`** section.
6. Select the **`Yes`** radio button to enable the Augeas lens.

## 4. Tracking Resource Configuration Changes

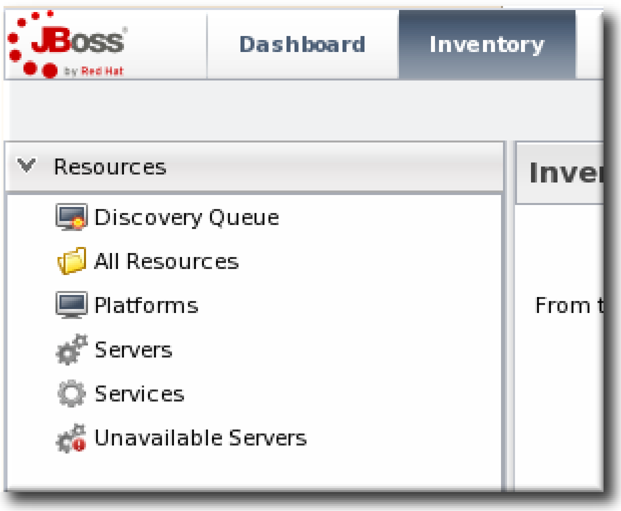
The revision numbers are global number across the JBoss ON server. For example, if Resource A is edited, then it gets revision #1. Then, when Resource B is edited, it gets revision #2, and the next edit gets #3.

**NOTE**

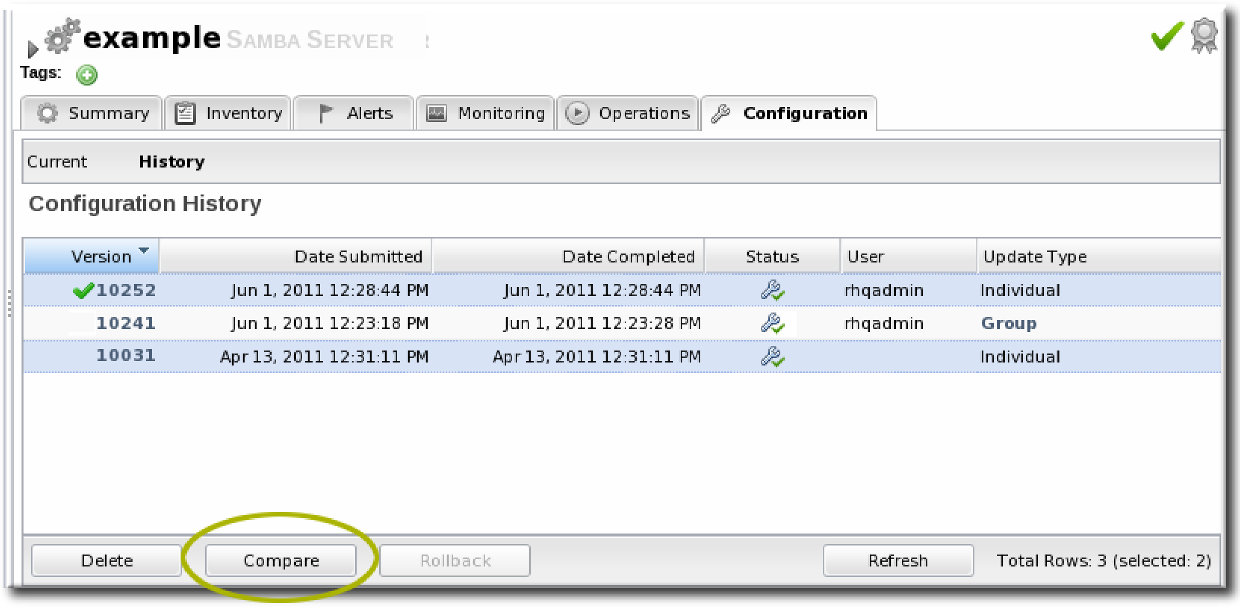
A user may have the right to edit or revert configuration, but that does not mean that the user has the right to delete an item from the configuration history.
Deleting elements in the history requires the manage inventory permission.

### 4.1. Tracking and Comparing Configuration Changes

1. Click the **`Inventory`** tab in the top menu.
2. Select the resource type in the **`Resources`** menu table on the left, and then browse or search for the resource.

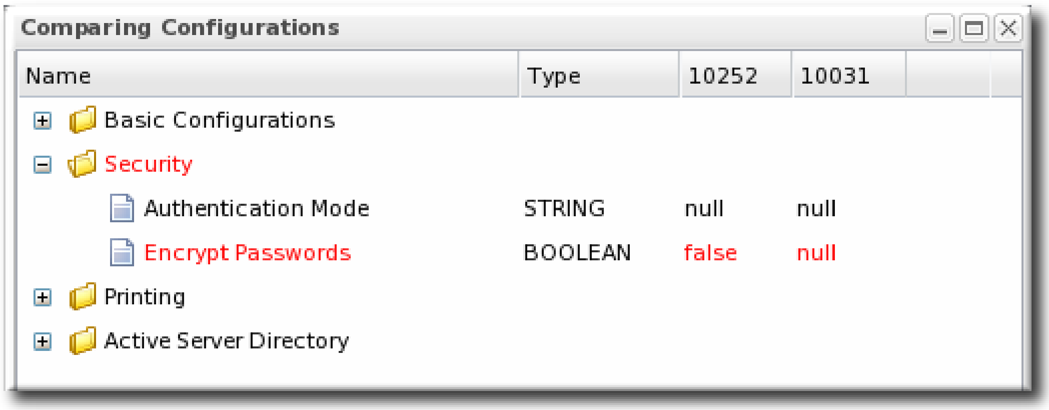


3. Open the **`Configuration`** tab for the resource.
4. Click the **`History`** subtab.
5. Select the line of the configuration version to view or compare. Use the **`Ctrl`** key to select multiple versions. The current (most recent successful) configuration state is marked by a green check mark.



| Version | Date Submitted | Date Completed | Status | User | Update Type |
|---|---|---|---|---|---|
| ✓10252 | Jun 1, 2011 12:28:44 PM | Jun 1, 2011 12:28:44 PM | | rhqadmin | Individual |
| 10241 | Jun 1, 2011 12:23:18 PM | Jun 1, 2011 12:23:28 PM | | rhqadmin | Group |
| 10031 | Apr 13, 2011 12:31:11 PM | Apr 13, 2011 12:31:11 PM | | | Individual |

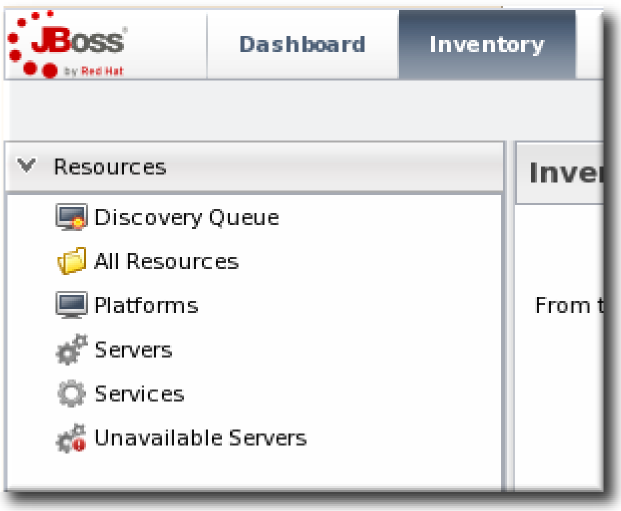
6. Click the **`Compare`** button.
7. The pop-up window shows all of the changes in a directory-style layout, with each of the configuration areas as a high-level directory. Any changes are marked in red, and the values are shown for each selected version.



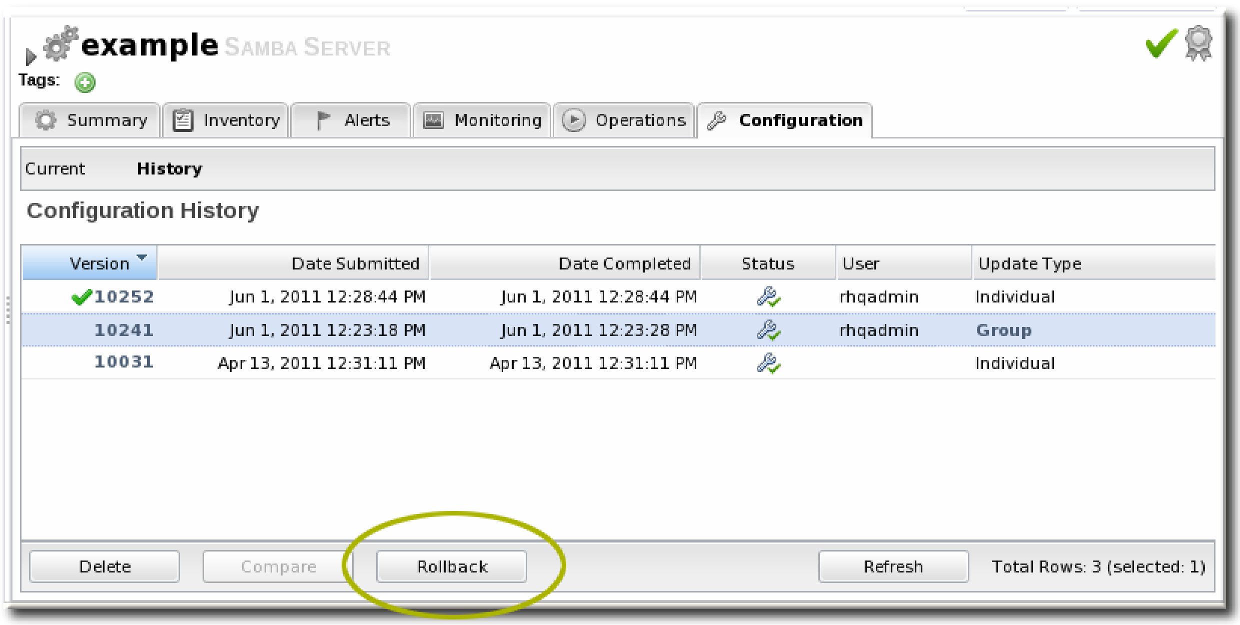
| Name | Type | 10252 | 10031 |
|---|---|---|---|
| Basic Configurations | | | |
| Security | | | |
| Authentication Mode | STRING | null | null |
| Encrypt Passwords | BOOLEAN | false | null |
| Printing | | | |
| Active Server Directory | | | |

## 4.2. Reverting Configuration Changes

1. Click the **`Inventory`** tab in the top menu.
2. Select the resource type in the **`Resources`** menu table on the left, and then browse or search for the resource.



3. Open the **`Configuration`** tab for the resource.
4. Click the **`History`** subtab.
5. Select the line of the configuration version to roll back to. The current (most recent successful) configuration state is marked by a green check mark.



| Version | Date Submitted | Date Completed | Status | User | Update Type |
|---|---|---|---|---|---|
| ✓10252 | Jun 1, 2011 12:28:44 PM | Jun 1, 2011 12:28:44 PM | ✓ | rhqadmin | Individual |
| 10241 | Jun 1, 2011 12:23:18 PM | Jun 1, 2011 12:23:28 PM | ✓ | rhqadmin | Group |
| 10031 | Apr 13, 2011 12:31:11 PM | Apr 13, 2011 12:31:11 PM | ✓ | | Individual |

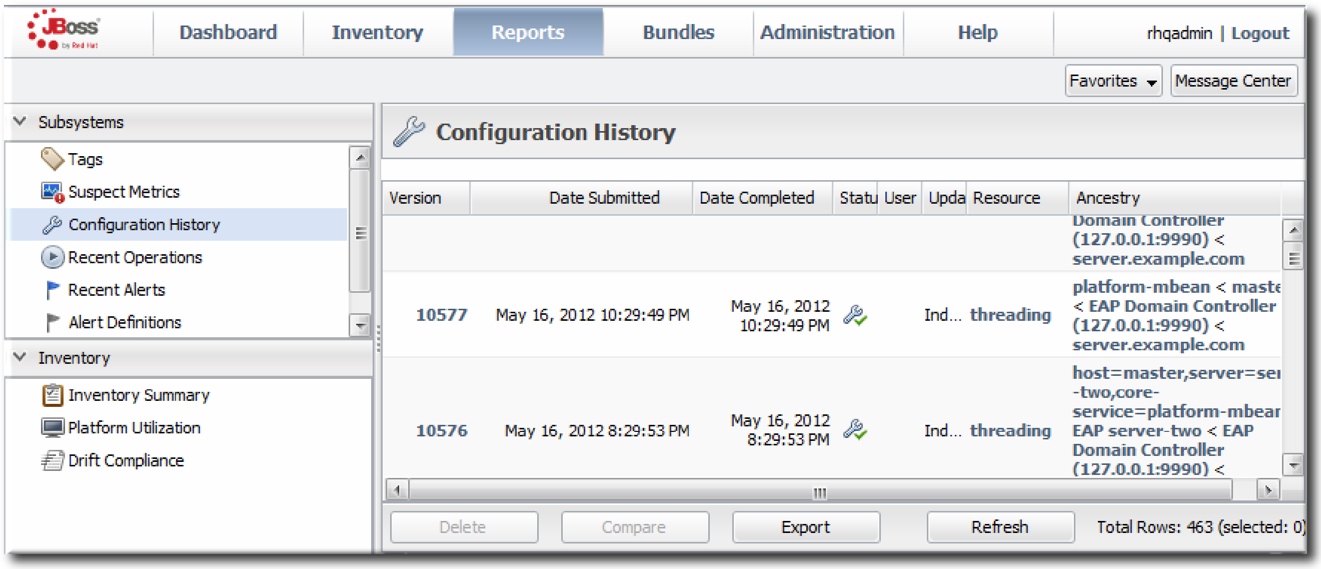
6. Click the **`Rollback`** button.

## 4.3. Viewing the Configuration History Report

Every resource (that supports configuration) tracks its own individual configuration change history, as in Section 4.1, “Tracking and Comparing Configuration Changes”.

JBoss ON also keeps a master list of all configuration changes, for all resources. This is displayed in the **`Configuration History Report`**, in the **`Reports`** tab.

As with the resource-level configuration history, the Configuration History shows the version number of the change, the time the configuration change was requested and completed, its status, and the requesting user. Because all resources are listed, the **`Configuration History Report`** also shows the resource name and its parent (and grandparent) to help disambiguate on which resource the change occurred.



| Version | Date Submitted | Date Completed | Statu | User | Upda | Resource | Ancestry |
|---|---|---|---|---|---|---|---|
| | | | | | | | Domain Controller (127.0.0.1:9990) < server.example.com |
| 10577 | May 16, 2012 10:29:49 PM | May 16, 2012 10:29:49 PM | | | Ind... | threading | platform-mbean < maste < EAP Domain Controller (127.0.0.1:9990) < server.example.com |
| 10576 | May 16, 2012 8:29:53 PM | May 16, 2012 8:29:53 PM | | | Ind... | threading | host=master,server=se -two,core-service=platform-mbear EAP server-two < EAP Domain Controller (127.0.0.1:9990) < |

Total Rows: 463 (selected: 0)

**Figure 3. Configuration History Report**

The **`Configuration History Report`** supports compare operations, as with the resource-level configuration history. This is useful because you can compare not only versions of configuration for the same resource, but also for the same configuration property in different resources (of the same type). This helps administrators figure out where their infrastructures are and to pinpoint changes or diversions between the configuration for similar resources.

**TIP**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.
To export a report, simply click the **`Export`** button. The report will automatically be downloaded as **`configurationHistory.csv`**.

## 5. Managing Configuration Drift

Much of the JBoss ON configuration management is designed around *implementing* changes for resources by editing configuration files or updating files and packages. But another aspect of managing configuration is *detecting changes*.

IT administrators must invest a significant amount of time planning the optimum configuration for systems in every type of environment, from production systems to internal resources. This ideal configuration includes file settings, software versions, and system settings. Resource configuration is going to change naturally over time, but administrators need to be able to track those changes to make sure that no unplanned or undesirable changes impact the resource. Defining a baseline configuration and tracking changes helps systems remain resilient during both maintenance and failures.

The unplanned changes that occur to a resource's configuration is called *drift*, as the configuration moves away from the designed baseline. Drift is common because of frequent software and hardware updates, particularly in a colocation facility or using virtual machines.

Production, staging, development, and recovery configurations are designed to have identical or near-identical configuration to maintain consistency. As the configurations within the different environments change, there emerges a *configuration gap*. Ultimately, this configuration gap can lead to disaster recovery failures or high availability failures because the configuration of the production system and the backup system are too different.

Drift monitoring provides a very general, freewheeling *content-based monitoring*. Rather than structured configuration management, drift monitoring tracks changes to content on the local filesystem. This means any changes in any files — even binary files [1].

## 5.1. Understanding Drift

Of course, drift monitoring isn't as simple as checking for changes. One of the core questions is *what changes matter?* There are two conceptual parts to that question:

- What directories (and files within those directories) matter for drift monitoring? Even though a drift definition is defined for a resource, the actual drift detection is performed at the directory level. Drift monitoring, then, can hit anywhere on a platform — even outside resources managed by JBoss ON.
- How do you identify a change? Do you compare it to the version immediately before it or to an established baseline?

Once you identify what changes you want to monitor for drift, then you can use JBoss ON to set up monitoring and alerting effectively.

### 5.1.1. Drift Definitions and Detection

The first part of drift detection is identifying what you are monitoring.

JBoss ON defines a *drift definition* that sets the target location for drift monitoring. The target can be identified from some configuration element for the resource — it can be a directory or file on the filesystem, a resource configuration property, a resource plug-in parameter, or a monitoring trait. This target is the *base directory*. The trait type is the *value context*, while the actual value is the *value name*. For example, for a base directory of **`/etc/`** that only includes changes to **`*.conf`** files, the elements in the drift definition are:

```
Value context: fileSystem
Value name: /etc
Includes: **/*.conf
```

**TIP**

Drift detection is performed at the filesystem level. This means that drift detection is not bound or restricted to a resource managed by JBoss ON. You can create a drift definition for a platform and set it to monitor any file or directory on that platform, even if it is outside the JBoss ON inventory, as long as that directory is accessible to the system user that the JBoss ON agent runs as.

By default, every subdirectory and file underneath the base directory is monitored for drift. The *includes/excludes* options define subdirectories or files that are explicitly included or explicitly excluded from drift monitoring. If *includes* is used, then only the specified directories or files are monitored and everything else is implicitly excluded, and vice versa for *excludes*. Included/excluded directories and files are identified by a path and a pattern. The path is the starting point beneath the base directory, and the pattern matches the file to monitor.

**Table 1. Combinations to Include Specific Files**

| Files to Monitor for Drift | 'Includes' Path | 'Includes' Pattern |
|---|---|---|
| /etc and all its subdirectories | Blank | Blank |
| For *.conf files in /etc and all subdirectories | . | **/*.conf [a] |
| For *.conf files only in the /etc directory, with no subdirectories (/etc/*.conf) | . | *.conf |
| For *.conf files only in a subdirectory one level below /etc (/etc/*/*.conf) | Not possible | Not possible |
| For any file in a specific subdirectory (yum.repos.d/) below /etc | yum.repos.d (subdirectory name) | Blank |

[a] This must have a double asterisk for the directory part. It will not work with a single asterisk.

The drift definition also sets an interval, or frequency, for how often the agent checks for drift. This is a very important setting for performance, both for JBoss ON and for data management. Setting a frequency that is too high risks missing changes or lumping changes together into large (and therefore difficult to manage) snapshots. However, setting the interval too low can impact JBoss ON agent and server performance.

**The key thing about the drift definition is that it sets what to look at and how often to look.**

**NOTE**

All drift detection runs are performed outside the agent plug-in and independent of the resource state. A drift detection scan can be run even if the resource is not running.

### 5.1.2. Snapshots, Deltas, and Baseline Images

The second part of drift detection is identifying how you want to define a *change*. Change is comparative. It takes the current version of a file and compares it to some previous version. The question for drift management is what previous version to compare to.

When a drift definition is first created, the agent collects all of the files in that base directory and subdirectories and sends information about them to the server. This collection is the initial file set.

From that point onward, the agent only sends change information about the files. Each set of changes is a *snapshot*. For text files, the change information includes the content of the file and diffs (both constructed on the JBoss ON server based on patches sent by the agent). For binary files, drift only records that the file change and displays a SHA and timestamp. A snapshot is always based on real files from a real resource.

**NOTE**

The agent does not send the actual files to the JBoss ON server. The agent sends information about file changes back to the server. These updates only contain the deltas between versions; they're not full files. This minimizes the network I/O.
The actual diffs are generated on the server from the content that the server stores.

The way that a snapshot is created is by comparing the current files against the agent's version. There are two ways that this comparison can be made:

- It can compare against the next-most recent version of the files.
- It can compare against a defined, stable baseline.

The first option is a *rolling snapshot*. This is the simplest setting, because it just keeps a running tally of changes.

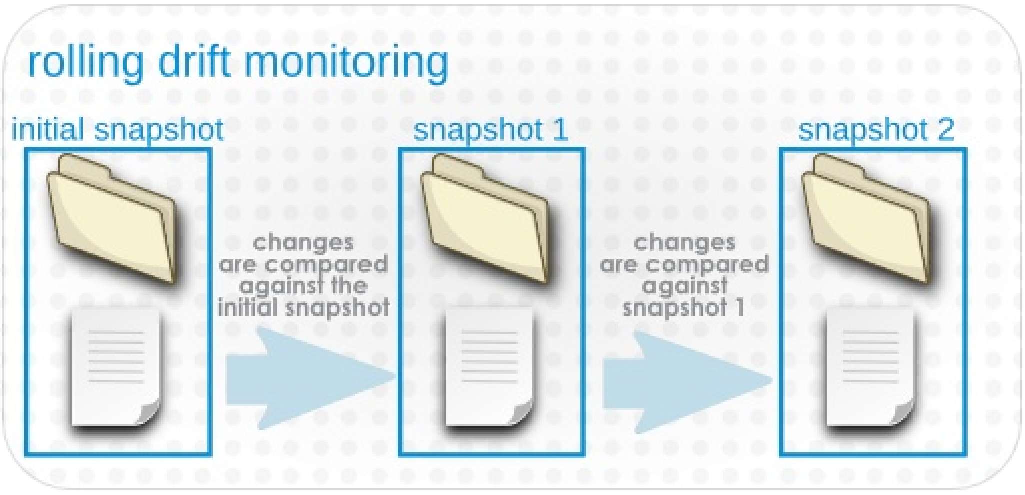

rolling drift monitoring
initial snapshot
snapshot 1
snapshot 2
changes are compared against the initial snapshot
changes are compared against snapshot 1

**Figure 4. Rolling Snapshots**

The second option is a *pinned snapshot*, and this is the method that gives administrators the most insight and control over drift. A pinned snapshot means that some image of the base directory has the optimal, approved configuration and this snapshot is selected as the baseline. It is in essence pinned in place, and every subsequent change is compared against that pinned snapshot, rather than being compared against each other.

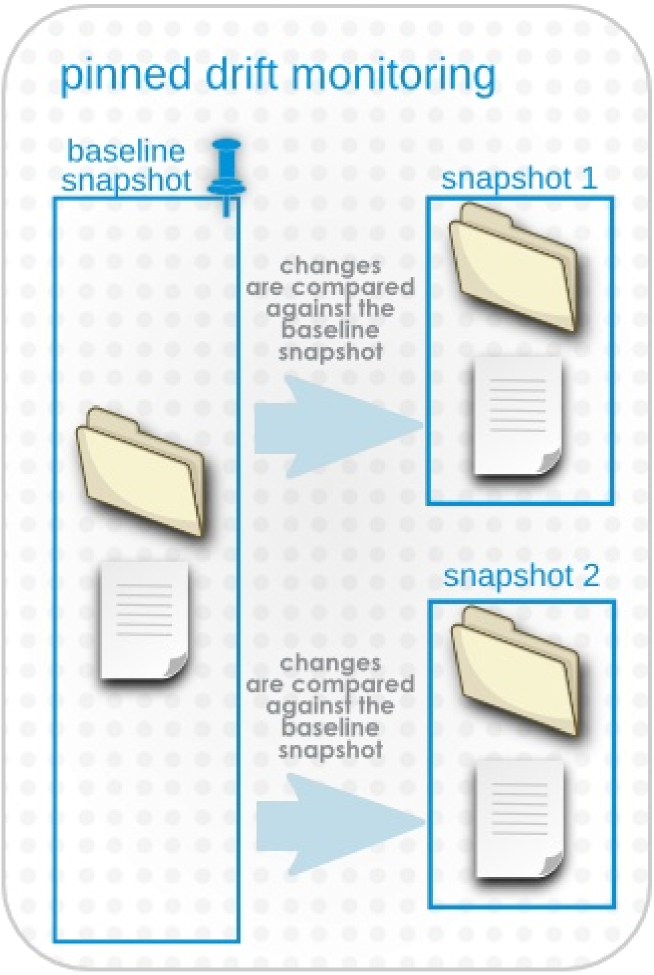

pinned drift monitoring
baseline snapshot
snapshot 1
changes are compared against the baseline snapshot
snapshot 2
changes are compared against the baseline snapshot

**Figure 5. Pinned Snapshots**

Snapshots exist at the resource-level, because they are based on the real files that exist on a system. When a snapshot is pinned to a resource-level definition, then any changes made on that system are compared to that snapshot. When the current file version matches the pinned snapshot, the resource is *compliant*.

A snapshot for a resource can also be pinned to a drift template — then, it is applied to every definition attached to that template. This is really powerful for administrators. For example, you can use a staging or development server to create the best system configuration for EAP performance, and then apply that EAP baseline snapshot to every EAP server in the production environment by pinning it to a drift template. You can see immediately all the EAP configuration relative to your defined ideal.

### 5.1.3. Destination Directories with Special File Types

Drift looks at both files and directories on the local system to generate snapshots and identify changes. The majority of these files and directories are going to be real files, but Unix does have some special file types, and the drift operation may encounter those files as part of processing the destination directory. There are some behavior implications, particularly with symbolic links and named pipes.

With symbolic links, drift detection follows any links back to the original file or directory, and includes those files in the snapshot. For example, if a symlink is set up for some library:

```
ln -ls /home/dev/libs /usr/share/jbossas/server/libs
```

If drift is configured on the **`libs/`** directory in the JBoss AS home directory, it will follow the symlink back to **`/home/dev/libs`**, and include all of those files in the drift snapshot.

**IMPORTANT**

Be careful when configuring drift against a directory which contains symlinks. All of the linked files will be included as part of the drift target.
If the linked directory has a large number of files, then drift detection runs may take longer than expected. Additionally, changes in that symlinked directory may have an unexpected impact on drift detection by recording many changes as drift when they weren't intended to be.
If you do not want to include the symlinked directory in the drift definition, use the **`excludes`** parameter in the drift definition to exclude the symlink.

The other special file type that is common on Unix systems is a *pipe*. As with a symlink, drift detection runs can detect a fifo file within the target directory. However, unlike symlinks, drift cannot process the fifo file, which causes drift detection to hang.

**NOTE**

Use the **`excludes`** parameter in the drift definition to exclude any named pipes in the target directory.

**Table 2. Drift Definitions and Unix File Types**

| File Type | Supported by Drift? |
|---|---|
| File | Yes |
| Directory | Yes |
| Symbolic link | Yes |
| Pipe | No |
| Socket | No |
| Device | No |

### 5.1.4. Drift and Resource Types

Whether drift is supported is defined in the resource type (which is discussed in Section 5.3.1, “About Resources and Drift Definition Templates”). If a drift template is defined in the resource type's **`rhq-plugin.xml`** descriptor, then that resource type supports drift. The template is a starting point (not an enforced configuration, like alert or metric collection templates).

Three JBoss ON standard resource types support drift:

- All platforms
- JBoss EAP 6 (AS 7), and all resources which use the JBoss AS 5 plug-in
- JBoss AS/EAP 5, and all resources which use the JBoss AS 5 plug-in
- JBoss AS/EAP 4 (deprecated)

Because drift support is defined in the plug-in descriptor, custom plug-ins can be created that add drift support for those resource types. For examples of writing agent plug-ins with drift support, see "Writing Custom JBoss ON Plug-ins."

**NOTE**

Drift is not supported on embedded web applications, such as an embedded WAR under an EAR application.

Drift detection is performed at the *directory level*. It is not tied to a specific resource. This means that drift detection can be run even when a resource is not running. It also means that drift detection can occur for an application, service, or file that is not managed by JBoss ON or for a resource type that does not otherwise support drift.

To monitor an entity outside the three supported resources, just configure drift detection on the platform resource and define the base directory as whatever directory path is used by the application or service you want to monitor.

### 5.1.5. Back to Drift Monitoring

The goal of setting up drift detection is to provide clarity into how systems and application servers are being modified. JBoss ON provides two ways to manage drift.

#### Drift Monitoring

Drift monitoring is the ability to track changes to target locations. The JBoss ON GUI allows you to view snapshots all together, compare changes for individual files between snapshots, view the current configuration, and view change details. It also provides inventory and drift reports and indicates, at a glance, whether a resource is compliant with an associated pinned snapshot.

#### Drift Alerting

A specific alert condition exists that will trigger an alert whenever there is drift. For rolling snapshots, this will send an alert once (and only once) for each drift snapshot. For pinned snapshots, the drift alert is fired for every detection run for as long as the resource is out of compliance, even if there are no subsequent changes.

**TIP**

There is no direct way to remedy drift through the JBoss ON GUI. However, it is possible to launch a JBoss ON CLI script in response to a drift alert. For example, you can create a patch of your ideal EAP configuration. If an EAP server drifts from that configuration, then you can use a JBoss ON CLI script to deploy that EAP patch bundle to the drifting EAP server.

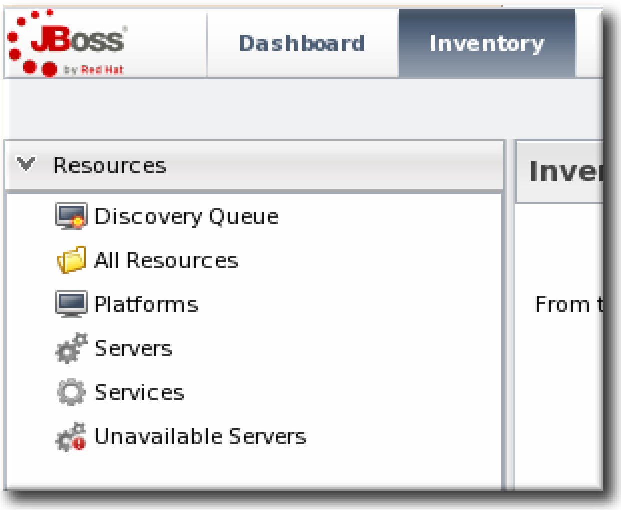
## 5.2. Adding a Drift Definition for a Resource

**IMPORTANT**

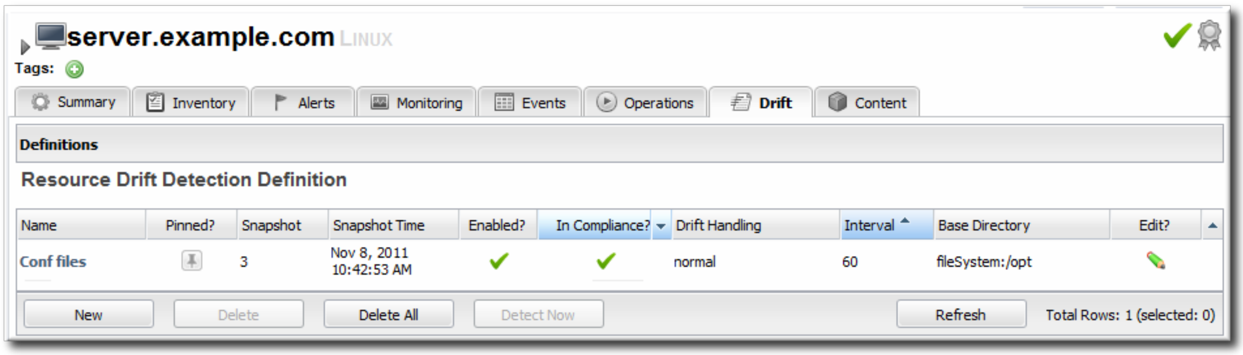
The directories where drift detection is being run *cannot* be changed after the definition is created. Be careful to get the base directory and the included and excluded files properly configured before saving.

1. Click the **`Inventory`** tab in the top menu.
2. Select the resource type in the **`Resources`** menu table on the left, and then browse or search for

the resource.



3. Open the **`Drift`** tab for the resource.



**Definitions**

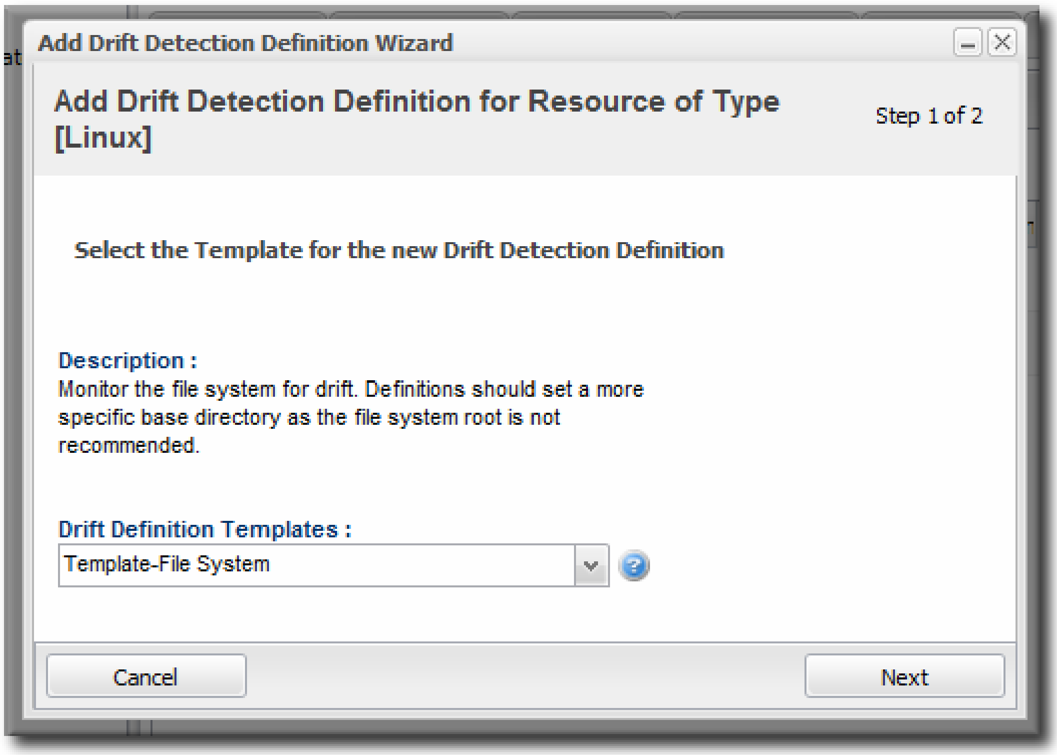
Resource Drift Detection Definition

| Name | Pinned? | Snapshot | Snapshot Time | Enabled? | In Compliance? | Drift Handling | Interval | Base Directory | Edit? |
|---|---|---|---|---|---|---|---|---|---|
| Conf files | | 3 | Nov 8, 2011 10:42:53 AM | ✓ | ✓ | normal | 60 | fileSystem:/opt | |

Total Rows: 1 (selected: 0)

4. Click the **`New`** at the bottom to add a new definition.
5. Select the template to use to as the basis for the new definition.

   Plug-in defined templates are defined in the platform and JBoss server resources, as well as any other resource which supports drift monitoring. Additional, user-defined templates can be also be created and applied.


Add Drift Detection Definition Wizard

Add Drift Detection Definition for Resource of Type [Linux]

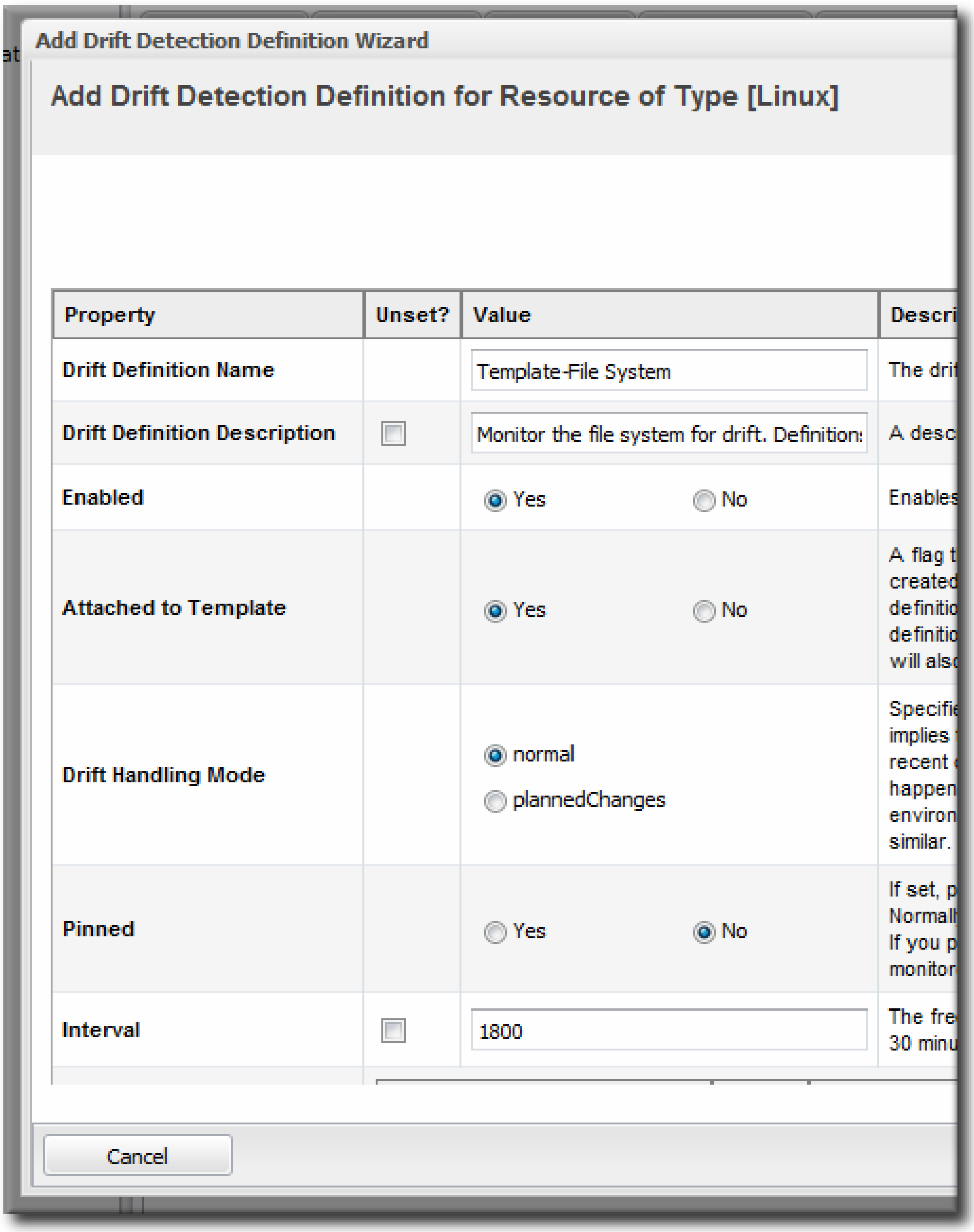
Step 1 of 2

Select the Template for the new Drift Detection Definition

Description :
Monitor the file system for drift. Definitions should set a more specific base directory as the file system root is not recommended.

Drift Definition Templates :
Template-File System

6. Give a unique name to the definition. The name and the base directory are combined to identify the definition within JBoss ON.


Add Drift Detection Definition Wizard

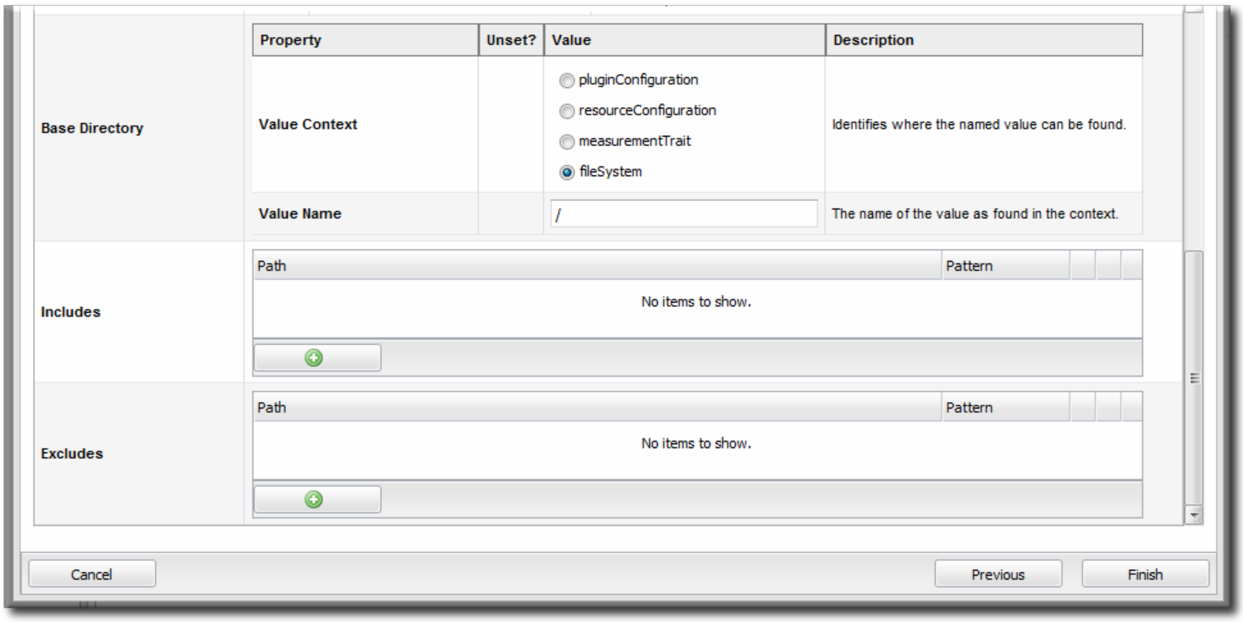
Add Drift Detection Definition for Resource of Type [Linux]

| Property | Unset? | Value | Descri |
|---|---|---|---|
| Drift Definition Name | | Template-File System | The drif |
| Drift Definition Description | ☐ | Monitor the file system for drift. Definition: | A desc |
| Enabled | | [✓ Yes] No | Enables |
| Attached to Template | | [✓ Yes] No | A flag t<br>created<br>definitio<br>definitio<br>will also |
| Drift Handling Mode | | [✓ normal]<br>plannedChanges | Specifie<br>implies t<br>recent c<br>happen<br>environ<br>similar. |
| Pinned | | Yes [✓ No] | If set, p<br>Normall<br>If you p<br>monitor |
| Interval | ☐ | 1800 | The fre<br>30 minu |

7. Define the settings for the definition, like the interval and whether it is associated with the template. The properties are listed in Table 3, "Drift Definition Properties".
8. Set the base directory. This is the top-most directory where drift detection is run for the definition, and the scan recourses down.

   The template itself defines an initial directory, but it may be useful to set a more specific directory to use.
9. Click the button with the green plus (+) sign to add a subdirectory to include or exclude. The directory can be the base directory by specifying a period (.) as the directory. The pattern identifies which files within the directory to recognize by the service, either to explicitly include or explicitly exclude.

   The filters support Ant-like FilePatterns, using a path and pattern. The patterns support asterisks (*) as wildcards for any number of characters and question marks (?) for single character wild cards. For example, `**/*.conf` can be used to include only `.conf` files in any subdirectory.



| | Property | Unset? | Value | Description |
|---|---|---|---|---|
| Base Directory | Value Context | | pluginConfiguration / resourceConfiguration / measurementTrait / [✓ fileSystem] | Identifies where the named value can be found. |
| | Value Name | | / | The name of the value as found in the context. |

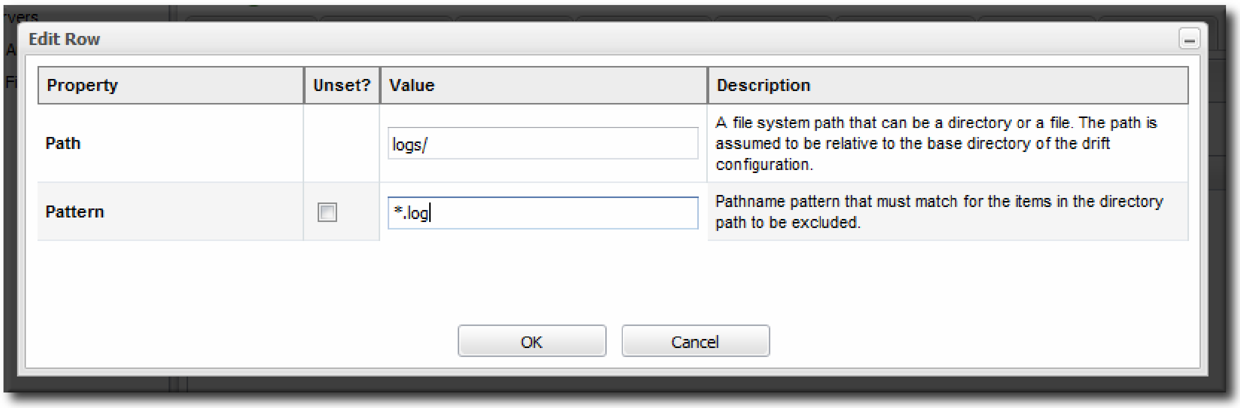
Includes

| Path | Pattern |
|---|---|
| No items to show. | |

Excludes

| Path | Pattern |
|---|---|
| No items to show. | |

There can be multiple include/exclude filters. Each directory and pattern can be added separately.



Edit Row

| Property | Unset? | Value | Description |
|---|---|---|---|
| Path | | logs/ | A file system path that can be a directory or a file. The path is assumed to be relative to the base directory of the drift configuration. |
| Pattern | ☐ | *.log | Pathname pattern that must match for the items in the directory path to be excluded. |

**Table 3. Drift Definition Properties**

| Property | Description |
|---|---|
| Name | A name for the drift detection definition. The name and the base directory, together, uniquely identify the definition. |
| Base Directory: Value Context | The type of configuration property which is used to identify the base directory. This identifies what type of element in the resource supplies the value. There are four options:<br>» File system, which is simply an absolute directory path on the resource. This directory must exist for drift to work.<br>» Resource configuration, which is a configuration property defined for the resource.<br>» Trait, which is one of the monitored traits for that resource.<br>» Plug-in configuration property, which is a property defined in the resource plug-in. |
| Base Directory: Value Name | The actual value for the drift detection definition to use for the base directory context. For example, if this is a file system context, then the value name is the directory path. |
| Includes | Explicitly includes directories, files, or files and directories matching a pattern, relative to the base directory, in the drift detection. The filters support Ant-like FilePatterns, using a path and pattern. The patterns support asterisks (*) as wildcards for any number of characters and question marks (?) for single character wild cards.<br>If a pattern is used, then a path must be specified, even if the path is the base directory. For example, to include only **`.conf`** files in the base directory, the pattern is **`*.conf`** and the path is a period (**`.`**) to indicate the local directory. |
| Excludes | Explicitly excludes directories, files, or files and directories matching a pattern, relative to the base directory, from the drift detection. The filters support Ant-like FilePatterns, using a path and pattern. The patterns support asterisks (*) as wildcards for any number of characters and question marks (?) for single character wild cards.<br>If a pattern is used, then a path must be specified, even if the path is the base directory. For example, to include only **`.conf`** files in the base directory, the pattern is **`*.conf`** and the path is a |

| | |
|---|---|
| | period (`.`) to indicate the local directory. |
| Enabled | Enables or disables the definition. Disabling a definition means that no detection scans are run. |
| Interval | Sets the frequency, in seconds, where the definition is eligible for a detection run. This is not a hard setting. Because load or other scheduled operations for the agent, the detection run is not guaranteed to run at the specified interval. |
| Pinned | Sets whether drift is determined in a rolling way or if it is associated (pinned) with a baseline snapshot. If this is set when the definition is created, then the initial snapshot is used as the baseline. Definitions attached to a pinned template cannot be unpinned. Definitions which are attached to an unpinned template or which are not attached to a template can be pinned or unpinned freely. |
| Drift Handling Mode | Sets whether drift changes are treated as events which trigger an alert (the default) or as expected, so that no alerts are triggered. |
| Attached to Template | Sets whether the resource-level definition is subordinate to a template. If it is attached to a template, then any changes to the template are reflected in the resource definition, including if the template is deleted. By default, definitions are attached to the template from which they are created. |
| Description | A simple text description of the definition. |

## 5.3. Creating a Drift Definition Template

Every time a new drift definition is created, it is based on an existing template for the resource type. At least one template is defined for resource types (by default, platforms and JBoss application servers) in their resource plug-in. Additional templates can be created by users.

### 5.3.1. About Resources and Drift Definition Templates

Resources of the same type frequently need to have the same, or similar, configuration settings. Particularly for an area like configuration drift, consistency is crucial for accurate and timely IT maintenance. JBoss ON allows this consistency using *drift definition templates*. Much like alert and monitoring templates, drift definition templates are defined for a resource *type* (regardless of whether any resources of that type actually exist) and can then be applied to specific resources in the inventory.

Drift templates are a little different than other template types in JBoss ON. First, a drift definition template is exactly that — it is an outline of default settings and values to use when creating a resource-level drift definition. It is not automatically applied to resources.

Additionally, there are two types of drift templates: plug-in defined templates and user-defined templates.

At least one drift definition template is actually defined as part of the plug-in for a resource type. Defining a template in the plug-in descriptor is what indicates that a resource type supports drift monitoring. These are *plug-in defined* templates; these are the default templates.

Having a plug-in defined template is the way that the JBoss ON agent recognizes that a particular resource type supports drift monitoring. So, the plug-in defined template has a dual purpose. It lets JBoss ON know what resource types support drift, and it gives basic input to help administrators start making their own drift definitions.

**Example 1. A JBoss Server Drift Definition Template**

```
<drift-definition name="Template-Base Files"
                  description="Monitor base application server files for drift.
It defines monitoring for some standard sub-directories of the HOME directory.
Note, it is not recommeded to monitor all files for an application server.
There are many files, and many temp files.">
    <basedir>
         <value-context>pluginConfiguration</value-context>
         <value-name>homeDir</value-name>
    </basedir>
    <includes>
 <include path="bin" />
        <include path="lib" />
        <include path="client" />
    </includes>
</drift-definition>
```

New drift definition templates can be added by administrators, in addition to the plug-in defined template; these are *user-defined* templates. These templates can reflect the unique infrastructure and application environment.

A resource-level drift definition is always based on a drift template, which provides some default values to the definition during creation. That template can be plug-in defined or user-defined. Resource-level drift definitions do not have to be attached to a template, so they do not have to be changed every time the template changes, but they are always based on an existing template.

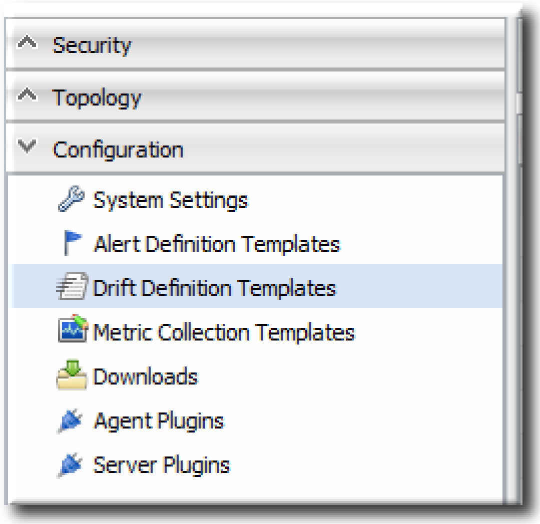
There are some things to remember about drift definition templates:

- Drift templates **are not** automatically applied to a resource, unlike other template types in JBoss ON. Drift templates are used as the basis for creating resource-level definitions.
- Default drift templates are defined for resources as part of their plug-in descriptor. Custom, user-defined templates can be added along with those defaults.
- Every drift definition is based on a template initially, even if that definition is not attached to that template post-creation.
- Snapshots (the file sets associated with drift definitions) always originate on a resource with a drift definition first. For any content to be associate with a template, the resource-level snapshot has to be promoted up to the template. Drift templates do not generate snapshots or files and then push that down to the resource.

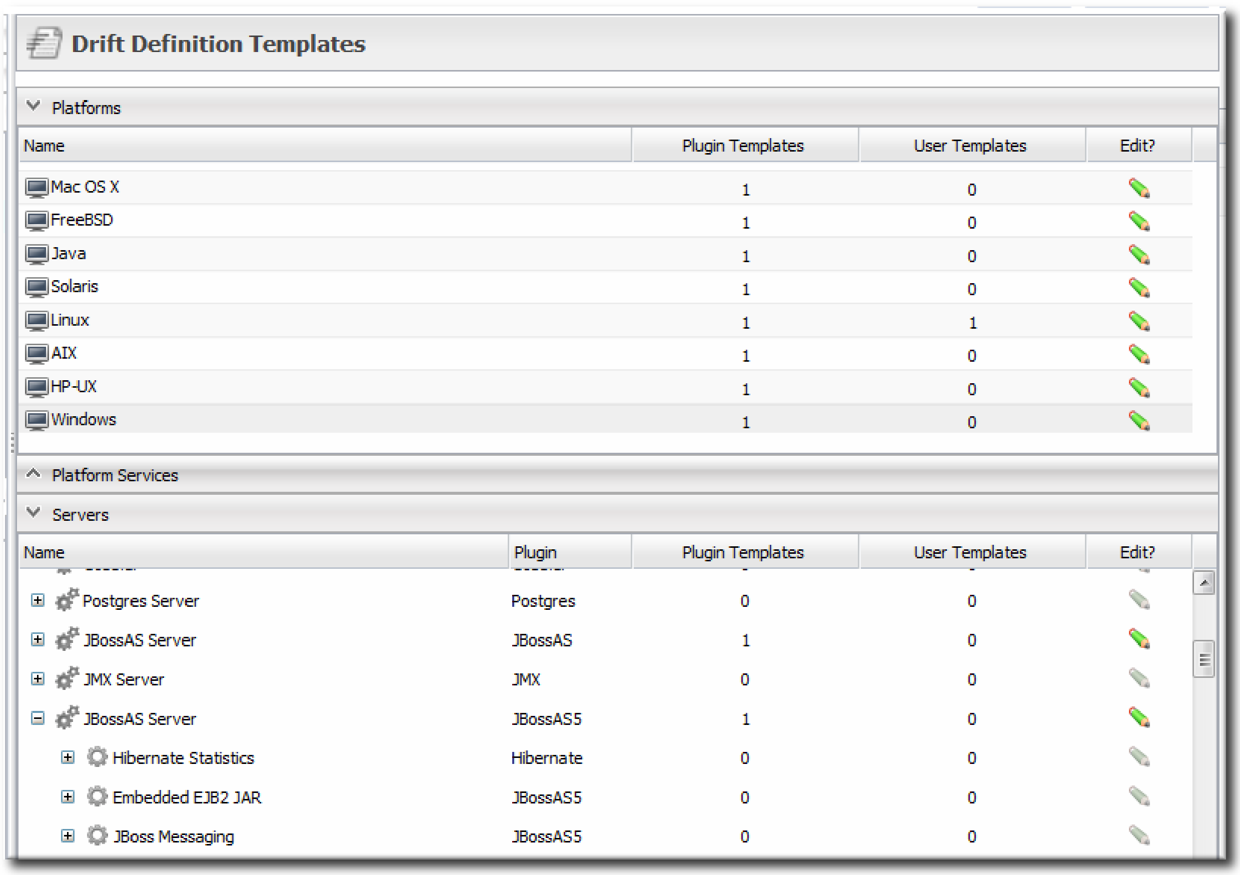
### 5.3.2. Creating a Drift Definition Template

A drift template creation form is almost identical to a resource-level drift definition, with two exceptions: it cannot be pinned to a snapshot at the time it is created and it cannot be associated with another template. Obviously, a template is not dependent on another template (even though it *is* created from another template.) Being unable to pin a template to a snapshot is also logical; when a template is created, it is not associated with any resources. So, it is not possible to generate snapshots, which means that there is nothing to pin the template to.

1. Click the **`Administration`** tab in the top menu.
2. Select the **`Drift Definition Templates`** menu table on the left.



3. Click the pencil icon for the resource type to add the template to. Not all resources support drift, so they cannot be selected.


Drift Definition Templates

Platforms

| Name | Plugin Templates | User Templates | Edit? |
|---|---|---|---|
| Mac OS X | 1 | 0 | ✎ |
| FreeBSD | 1 | 0 | ✎ |
| Java | 1 | 0 | ✎ |
| Solaris | 1 | 0 | ✎ |
| Linux | 1 | 1 | ✎ |
| AIX | 1 | 0 | ✎ |
| HP-UX | 1 | 0 | ✎ |
| Windows | 1 | 0 | ✎ |

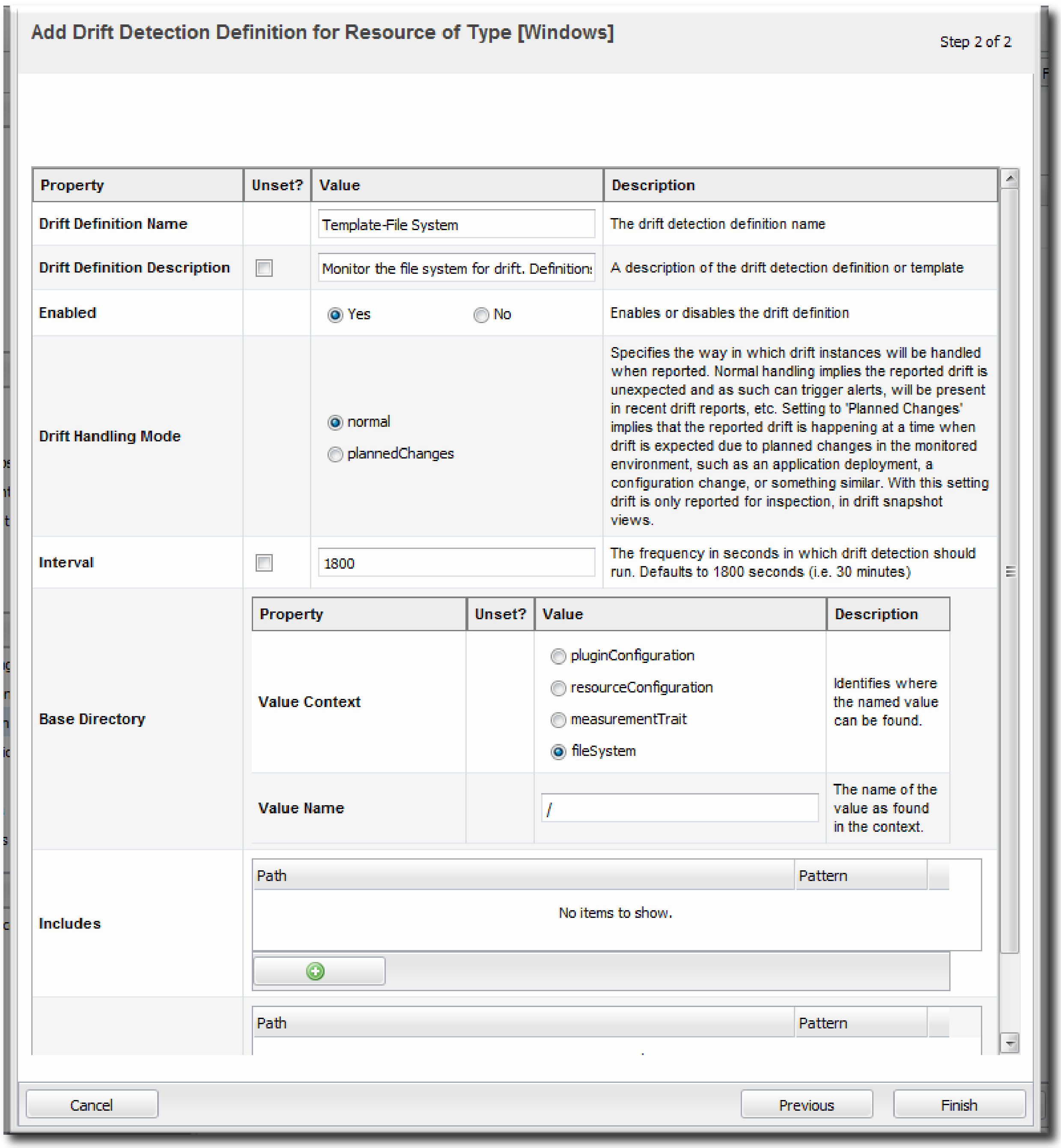
Platform Services

Servers

| Name | Plugin | Plugin Templates | User Templates | Edit? |
|---|---|---|---|---|
| ⊞ Postgres Server | Postgres | 0 | 0 | ✎ |
| ⊞ JBossAS Server | JBossAS | 1 | 0 | ✎ |
| ⊞ JMX Server | JMX | 0 | 0 | ✎ |
| ⊟ JBossAS Server | JBossAS5 | 1 | 0 | ✎ |
| ⊞ Hibernate Statistics | Hibernate | 0 | 0 | ✎ |
| ⊞ Embedded EJB2 JAR | JBossAS5 | 0 | 0 | ✎ |
| ⊞ JBoss Messaging | JBossAS5 | 0 | 0 | ✎ |

4. Click the **`New`** at the bottom to add a new template.
5. Select the template to use to as the basis for the new template.

   Plug-in defined templates are defined in the platform and JBoss server resources, as well as any other resource which supports drift monitoring. Additional, user-defined templates can be also be created and applied.
6. Give a unique name to the template. The name and the base directory are combined to identify the definition within JBoss ON.



Add Drift Detection Definition for Resource of Type [Windows]

Step 2 of 2

| Property | Unset? | Value | Description |
|---|---|---|---|
| Drift Definition Name | | Template-File System | The drift detection definition name |
| Drift Definition Description | ☐ | Monitor the file system for drift. Definition: | A description of the drift detection definition or template |
| Enabled | | [✓ Yes] No | Enables or disables the drift definition |
| Drift Handling Mode | | [✓ normal] plannedChanges | Specifies the way in which drift instances will be handled when reported. Normal handling implies the reported drift is unexpected and as such can trigger alerts, will be present in recent drift reports, etc. Setting to 'Planned Changes' implies that the reported drift is happening at a time when drift is expected due to planned changes in the monitored environment, such as an application deployment, a configuration change, or something similar. With this setting drift is only reported for inspection, in drift snapshot views. |
| Interval | ☐ | 1800 | The frequency in seconds in which drift detection should run. Defaults to 1800 seconds (i.e. 30 minutes) |
| Base Directory | | | |

| Property | Unset? | Value | Description |
|---|---|---|---|
| Value Context | | pluginConfiguration resourceConfiguration measurementTrait [✓ fileSystem] | Identifies where the named value can be found. |
| Value Name | | / | The name of the value as found in the context. |

Includes

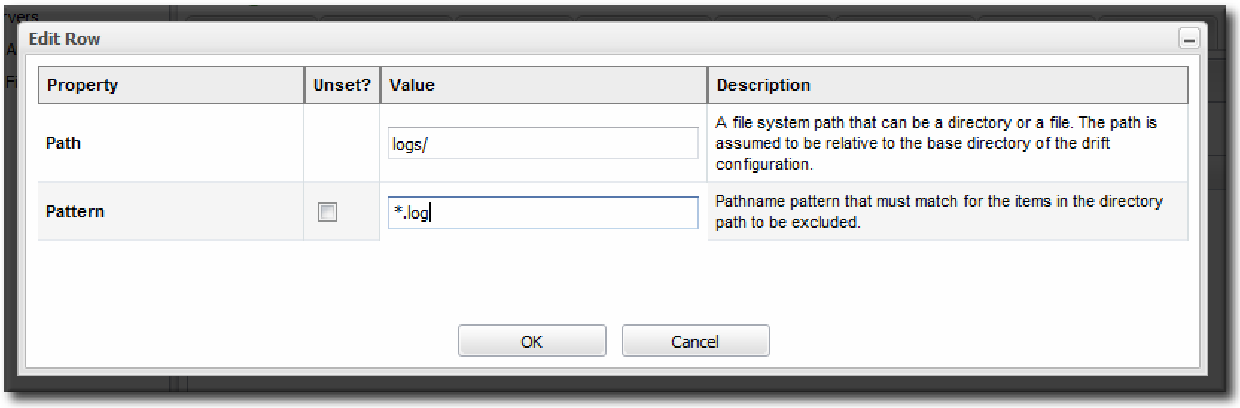
| Path | Pattern |
|---|---|
| No items to show. | |

| Path | Pattern |
|---|---|

Cancel Previous Finish

7. Define the settings for the definition, like the interval and whether it is enabled by default. The properties are listed in Table 3, "Drift Definition Properties".
8. Set the base directory. This is the top-most directory where drift detection is run for the definition, and the scan recourses down.
9. Click the button with the green plus (+) sign to add a subdirectory to include or exclude. The directory can be the base directory by specifying a period (.) as the directory. The pattern identifies which files within the directory to recognize by the service, either to explicitly include or explicitly exclude.

   The filters support Ant-like FilePatterns, using a path and pattern. The patterns support asterisks (*) as wildcards for any number of characters and question marks (?) for single character wild cards. For example, `**/*.conf` can be used to include only `.conf` files in any subdirectory.

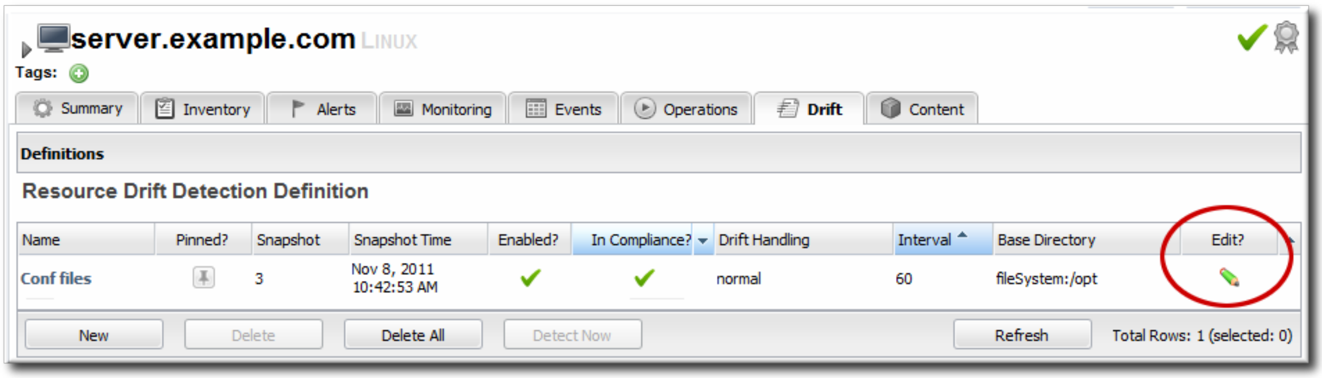

Edit Row

| Property | Unset? | Value | Description |
|---|---|---|---|
| Path | | logs/ | A file system path that can be a directory or a file. The path is assumed to be relative to the base directory of the drift configuration. |
| Pattern | ☐ | *.log | Pathname pattern that must match for the items in the directory path to be excluded. |

## 5.4. Editing Drift Definitions

Most entries in JBoss ON are edited by clicking their name or double-clicking their row in a list. However, for drift definitions, clicking the name or double-clicking the row opens up the list of snapshots for that definition — not the definition entry itself.

To edit a drift detection definition, click the pencil icon.



Resource Drift Detection Definition

| Name | Pinned? | Snapshot | Snapshot Time | Enabled? | In Compliance? | Drift Handling | Interval | Base Directory | Edit? |
|---|---|---|---|---|---|---|---|---|---|
| Conf files | | 3 | Nov 8, 2011 10:42:53 AM | ✓ | ✓ | normal | 60 | fileSystem:/opt | |

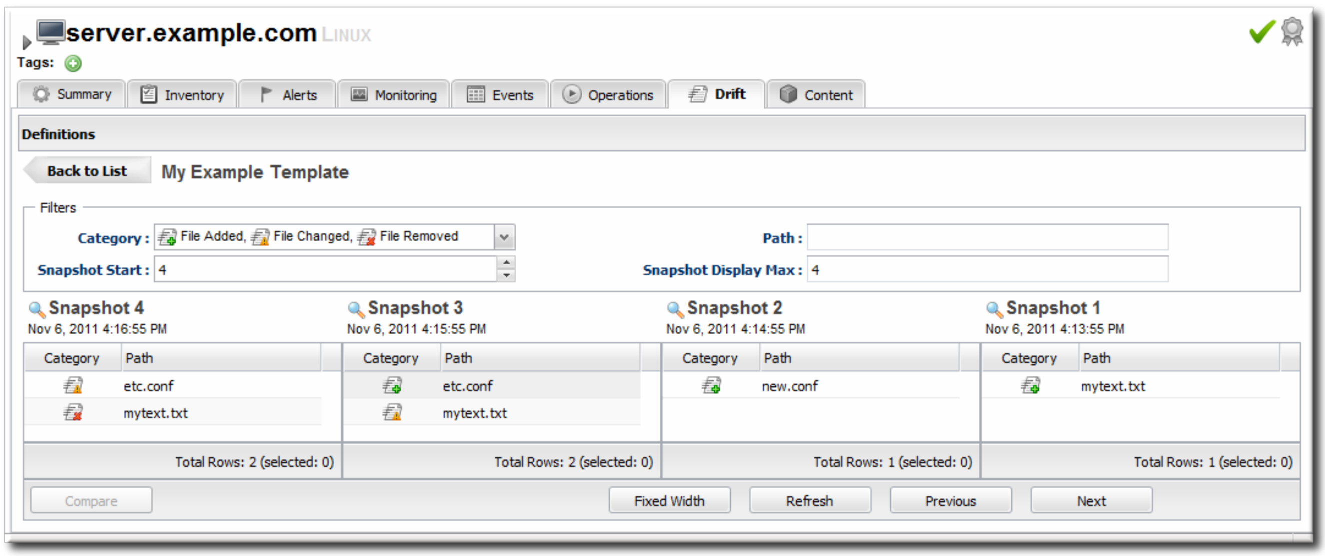
## 5.5. Viewing Snapshots and Changes

**NOTE**

The initial snapshot is *snapshot 0*. The snapshots in the carousel begin at version 1 — meaning it begins at the first change, not the initial file set.
If a snapshot is pinned, so that it is set as a baseline, then it is not displayed in the carousel because it is snapshot 0. However, it can be viewed by clicking the pinned icon in the definition list.

### 5.5.1. Viewing the Snapshot Carousel

Snapshots for a drift definition are displayed in a horizontal stream of windows, starting with the most recent change. This is colloquially called a *carousel*, because it is a rotating view of snapshots.



My Example Template

Filters

Category : File Added, File Changed, File Removed

Path :

Snapshot Start : 4

Snapshot Display Max : 4

Snapshot 4
Nov 6, 2011 4:16:55 PM

| Category | Path |
|---|---|
| File Changed | etc.conf |
| File Removed | mytext.txt |

Total Rows: 2 (selected: 0)

Snapshot 3
Nov 6, 2011 4:15:55 PM

| Category | Path |
|---|---|
| File Changed | etc.conf |
| File Changed | mytext.txt |

Total Rows: 2 (selected: 0)

Snapshot 2
Nov 6, 2011 4:14:55 PM

| Category | Path |
|---|---|
| File Added | new.conf |

Total Rows: 1 (selected: 0)

Snapshot 1
Nov 6, 2011 4:13:55 PM

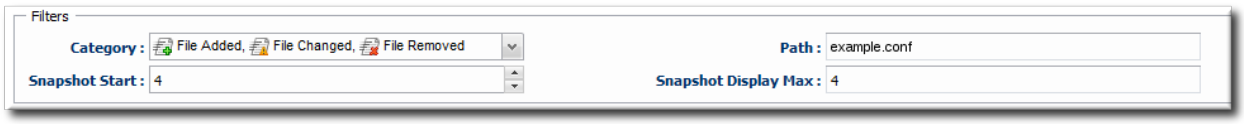
| Category | Path |
|---|---|
| File Added | mytext.txt |

Total Rows: 1 (selected: 0)

**Figure 6. Viewing Snapshots**

To open the carousel:

1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab for the resource.
4. Click the name of the drift definition.
5. The snapshot carousel shows, by default, the four most recent snapshots.
6. Optionally, filter the snapshots to view. There are two elements that can be used to search for snapshots:
   - The change type within the snapshot, whether a file was added, deleted, or modified.
   - The path of a change within the snapshot. This path filter is a substring filter based on the paths and files in the drift entries.



Filters

Category : File Added, File Changed, File Removed

Path : example.conf

Snapshot Start : 4

Snapshot Display Max : 4

There can be slight differences in the way that changes are recorded in snapshots if the definition is pinned. The most obvious is that if a new file is added, it will show up as a new file in every subsequent snapshot because it is always compared against the pinned snapshot, where the file does not exist. Likewise, if a file is deleted, it is listed in every snapshot as deleted.

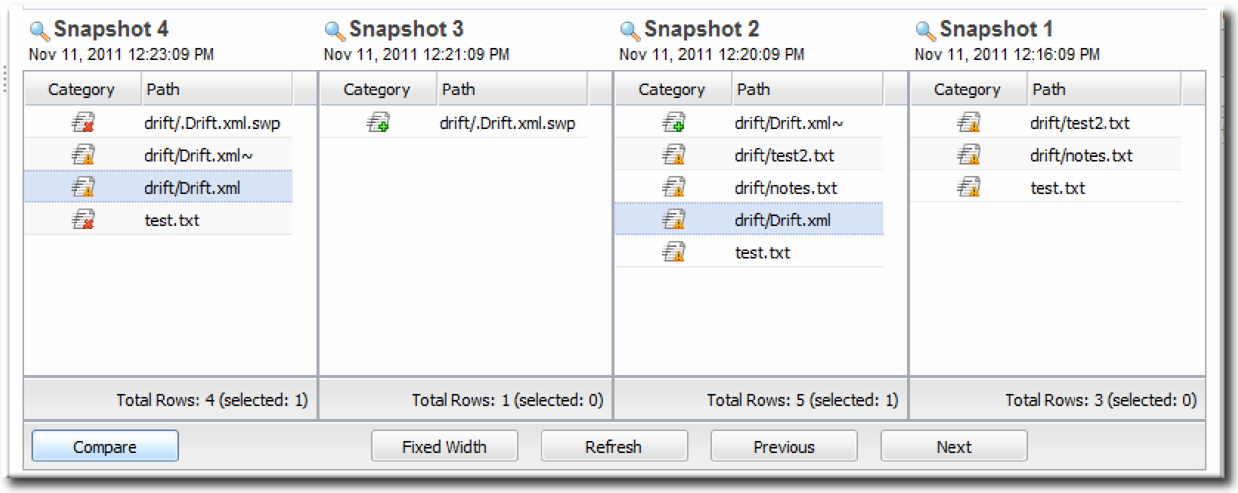
### 5.5.2. Comparing Drift Changes

Changes are diffed at the file level, not the full snapshot level. Administrators can view the specific changes made between versions on the selected files.

**NOTE**

Only changes for text files can be compared. Drift detection will identify binary files that have changed and show a timestamp and SHA, but it does not display the binary file contents or diff changes between versions of a binary file.

1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab for the resource.
4. Click the name of the drift definition.
5. Click the names of the files to compare.



**Snapshot 4** — Nov 11, 2011 12:23:09 PM

| Category | Path |
|---|---|
| | drift/.Drift.xml.swp |
| | drift/Drift.xml~ |
| | drift/Drift.xml |
| | test.txt |

Total Rows: 4 (selected: 1)

**Snapshot 3** — Nov 11, 2011 12:21:09 PM

| Category | Path |
|---|---|
| | drift/.Drift.xml.swp |

Total Rows: 1 (selected: 0)

**Snapshot 2** — Nov 11, 2011 12:20:09 PM

| Category | Path |
|---|---|
| | drift/Drift.xml~ |
| | drift/test2.txt |
| | drift/notes.txt |
| | drift/Drift.xml |
| | test.txt |

Total Rows: 5 (selected: 1)

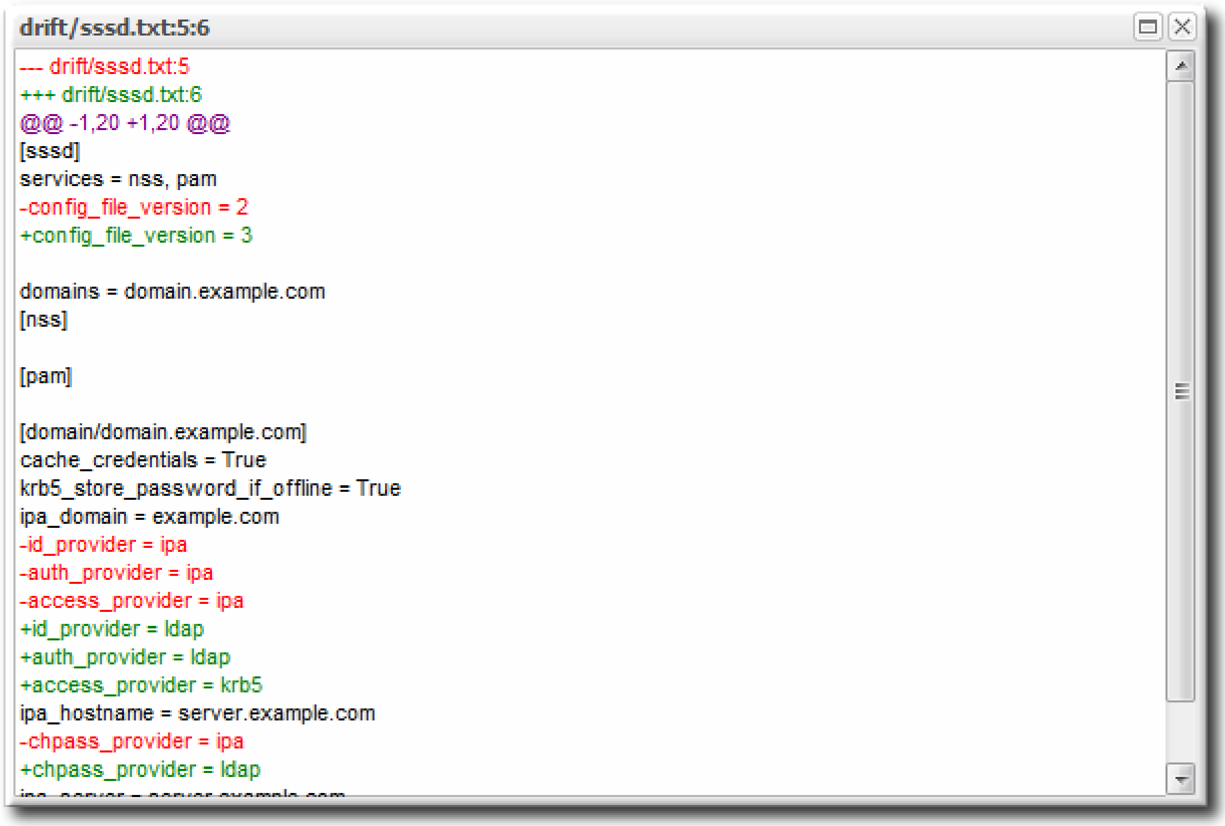
**Snapshot 1** — Nov 11, 2011 12:16:09 PM

| Category | Path |
|---|---|
| | drift/test2.txt |
| | drift/notes.txt |
| | test.txt |

Total Rows: 3 (selected: 0)

6. Click **`Compare`**.

The diff uses standard text formatting for displaying file diffs.



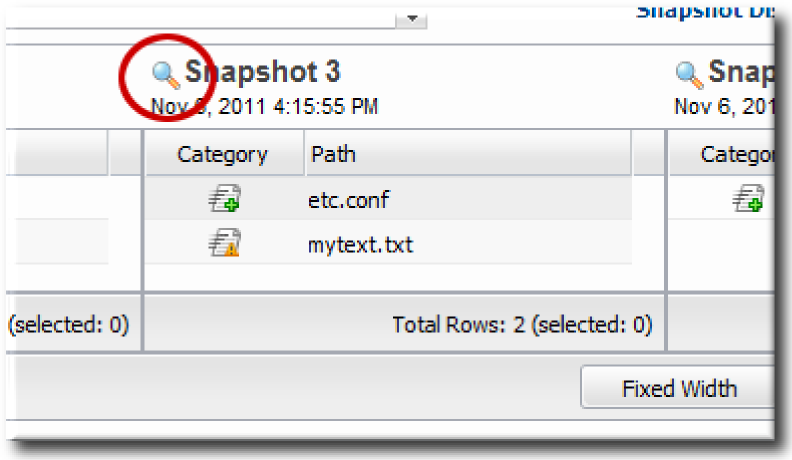
drift/sssd.txt:5:6

```
--- drift/sssd.txt:5
+++ drift/sssd.txt:6
@@ -1,20 +1,20 @@
[sssd]
services = nss, pam
-config_file_version = 2
+config_file_version = 3

domains = domain.example.com
[nss]

[pam]

[domain/domain.example.com]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example.com
-id_provider = ipa
-auth_provider = ipa
-access_provider = ipa
+id_provider = ldap
+auth_provider = ldap
+access_provider = krb5
ipa_hostname = server.example.com
-chpass_provider = ipa
+chpass_provider = ldap
```

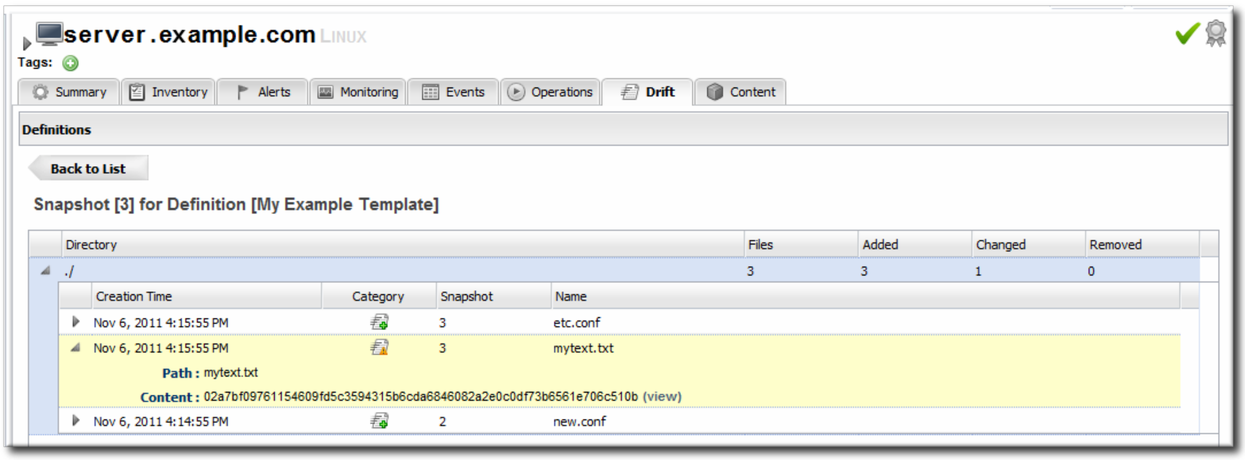
**Figure 7. Change Set Diffs**

## 5.5.3. Viewing Snapshot Details

1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab for the resource.
4. Click the name of the drift definition.
5. In the snapshot carousel, click the magnifying glass by the name of the snapshot to view.



6. Expand the directory to show the list of changes for that snapshot.



Snapshot [3] for Definition [My Example Template]

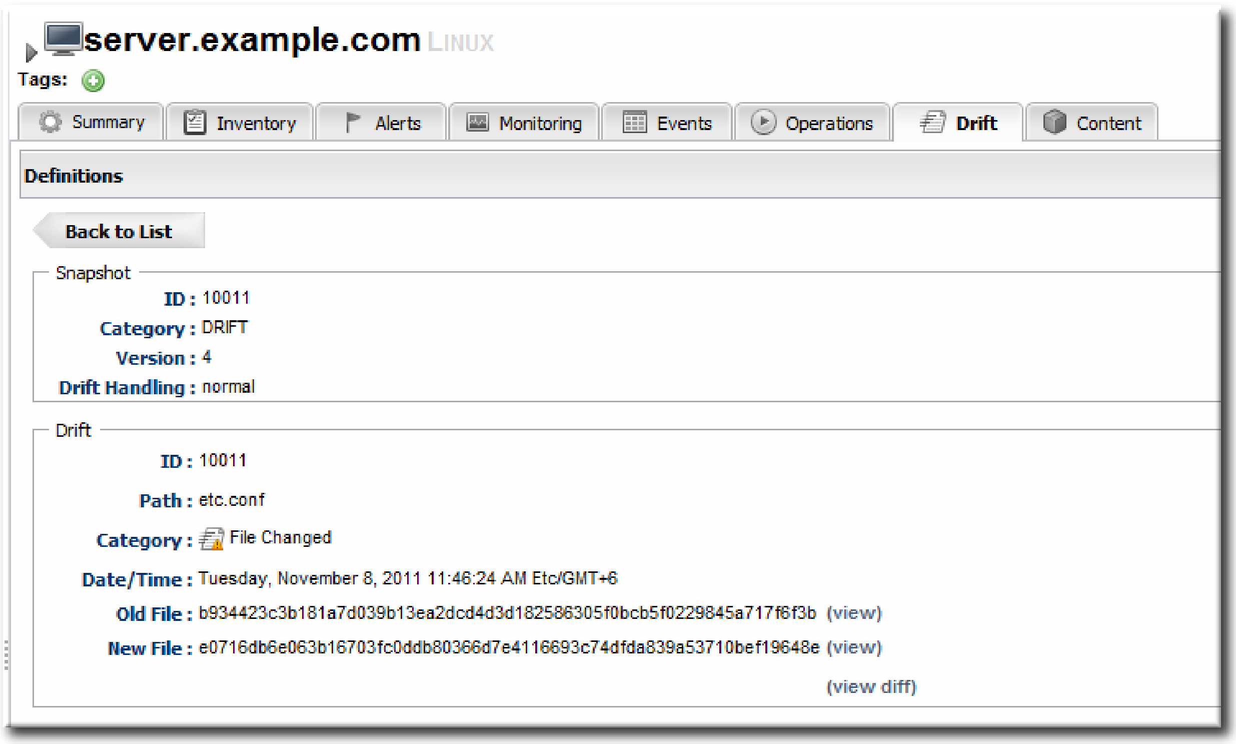
| Directory | Files | Added | Changed | Removed |
|---|---|---|---|---|
| ./ | 3 | 3 | 1 | 0 |

| Creation Time | Category | Snapshot | Name |
|---|---|---|---|
| Nov 6, 2011 4:15:55 PM | | 3 | etc.conf |
| Nov 6, 2011 4:15:55 PM | | 3 | mytext.txt |
| Nov 6, 2011 4:14:55 PM | | 2 | new.conf |

**Path :** mytext.txt

**Content :** 02a7bf09761154609fd5c3594315b6cda6846082a2e0c0df73b6561e706c510b (view)

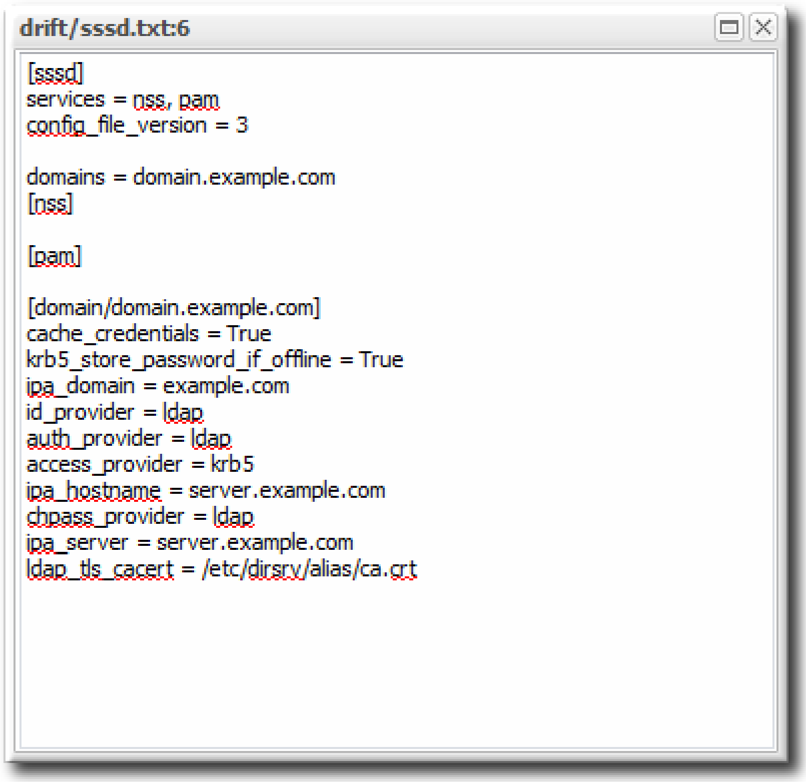
7. To see the details of a specific change, click the **`(view)`** link.
8. The details for that file shows links to display the immediate previous version of the file, the changed version of the file, and a diff between the two.



Snapshot
ID : 10011
Category : DRIFT
Version : 4
Drift Handling : normal

Drift
ID : 10011
Path : etc.conf
Category : File Changed
Date/Time : Tuesday, November 8, 2011 11:46:24 AM Etc/GMT+6
Old File : b934423c3b181a7d039b13ea2dcd4d3d182586305f0bcb5f0229845a717f6f3b (view)
New File : e0716db6e063b16703fc0ddb80366d7e4116693c74dfda839a53710bef19648e (view)
(view diff)

When clicking the **view** link, the page title has the version number along with the file name. For example, when viewing version 6 of **myfile.txt**, the title is *myfile.txt:6*.


drift/sssd.txt:6

```
[sssd]
services = nss, pam
config_file_version = 3

domains = domain.example.com
[nss]

[pam]

[domain/domain.example.com]
cache_credentials = True
krb5_store_password_if_offline = True
ipa_domain = example.com
id_provider = ldap
auth_provider = ldap
access_provider = krb5
ipa_hostname = server.example.com
chpass_provider = ldap
ipa_server = server.example.com
ldap_tls_cacert = /etc/dirsrv/alias/ca.crt
```

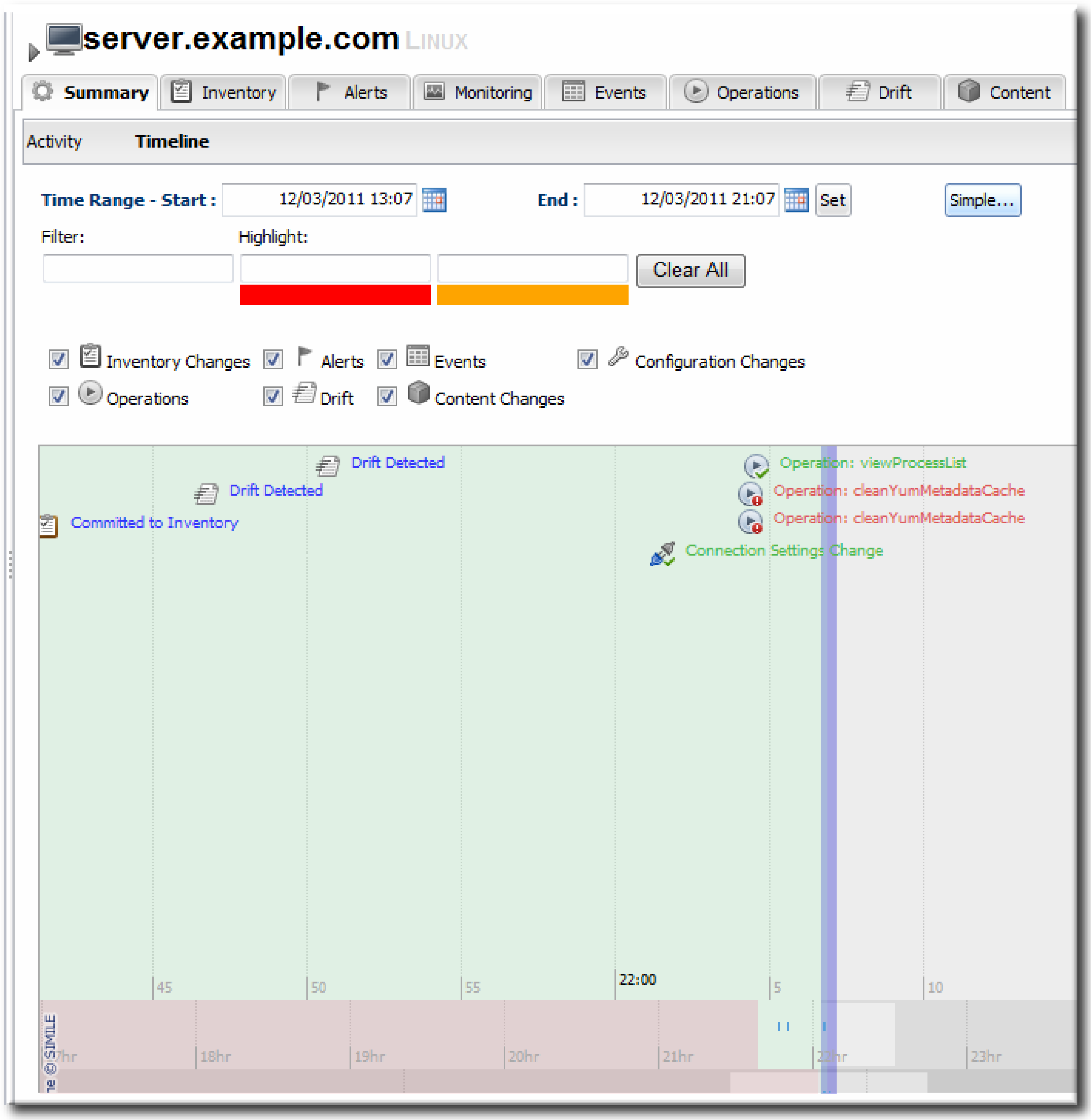
### 5.5.4. Seeing Drift Events in the Timeline

Whenever drift is detected, it shows up as an event in the events timeline for the resource.

1. Click the **Inventory** tab in the top menu.
2. Search for the resource.
3. In the **Summary** tab, click the **Timeline** subtab.

4. The detection runs where drift was detected show up in the timeline as **`Drift Detected`**. To see only drift events in the timeline, clear all but the **`Drift`** checkbox.

   The time interval can be reset to adjust the span of the timeline.



Time Range - Start : 12/03/2011 13:07 End : 12/03/2011 21:07

Filter:

Highlight:

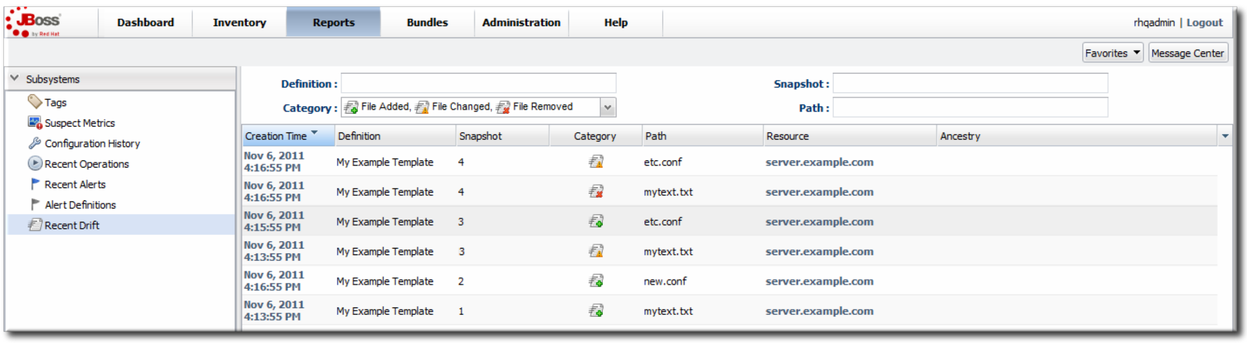
☑ Inventory Changes ☑ Alerts ☑ Events ☑ Configuration Changes
☑ Operations ☑ Drift ☑ Content Changes

Drift Detected
Drift Detected
Committed to Inventory
Operation: viewProcessList
Operation: cleanYumMetadataCache
Operation: cleanYumMetadataCache
Connection Settings Change

## 5.5.5. Checking Drift Snapshot Reports

The snapshot carousel (Section 5.5.1, “Viewing the Snapshot Carousel”) shows all of the snapshots for a single drift definition on a single resource. To view a list of all snapshots, for all definitions across all resources, check the Recent Drift Report.

1. Click the **`Reports`** tab in the top navigation menu.
2. Select the **`Recent Drift`** report from the **`Subsystems`** report list.
3. Every drift instance is listed, sorted by the snapshot creation time.



Definition :
Category : File Added, File Changed, File Removed
Snapshot :
Path :

| Creation Time | Definition | Snapshot | Category | Path | Resource | Ancestry |
|---|---|---|---|---|---|---|
| Nov 6, 2011 4:16:55 PM | My Example Template | 4 | ! | etc.conf | server.example.com | |
| Nov 6, 2011 4:16:55 PM | My Example Template | 4 | ✗ | mytext.txt | server.example.com | |
| Nov 6, 2011 4:15:55 PM | My Example Template | 3 | + | etc.conf | server.example.com | |
| Nov 6, 2011 4:13:55 PM | My Example Template | 3 | ! | mytext.txt | server.example.com | |
| Nov 6, 2011 4:16:55 PM | My Example Template | 2 | + | new.conf | server.example.com | |
| Nov 6, 2011 4:13:55 PM | My Example Template | 1 | + | mytext.txt | server.example.com | |

4. Optionally, filter the list of drift changes. There are four filter options:
   - The definition name
   - The snapshot number (which crosses drift definitions)
   - The change type within the snapshot, whether a file was added, deleted, or modified.
   - The path of a change within the snapshot. This path can be a directory, a specific file name, or a search expression.

**TIP**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.
Only the information displayed for the report is exported. If the **`Recent Drift Report`** is filtered by date, definition, snapshot or version, or category, only the matching operations are included in the report.
To export a report, simply click the **`Export`** button. The report will automatically be downloaded as **`recentDrift.csv`**.

## 5.6. Pinning Snapshots and Managing Compliance

As discussed in Section 5.1.2, “Snapshots, Deltas, and Baseline Images”, a specific snapshot, with its complete current fileset, can be associated or *pinned* to a drift definition. Pinning a snapshot creates an entirely new style of drift definition. Rather than simply tracking changes, a pinned snapshot allows an administrator to establish a clear, blessed configuration for a system or application. It sets a standard with which the system configuration should comply.

### 5.6.1. More About Pinning Snapshots

A snapshot is a picture of the actual, current files that are on a specific resource. A snapshot is a real-world view. In normal drift conditions, each snapshot is compared to the one immediately before it to show changes. However, it is possible to select a specific snapshot as a fixed baseline to compare changes against. This is a *pinned snapshot*.

A drift definition sets the rules for running drift detection, but it does not add or define or overwrite any files on a resource. A drift definition does not define content or contain a file set. Content has to be added to a definition (or a definition template). A file set (a snapshot) has to be manually added to the drift definition, after the snapshot exists. This is pinning. Pinning takes a real, existing set of files from a snapshot and links it to a drift definition on a resource or a drift definition template.

Pinning is one method that administrators can use to standardize resource configuration. An administrator can use a single resource as a test box to get a resource's configuration tuned to its ideal settings. Then, that file set can be pinned to a template and re-applied to other resources of the same type. Because the pinned snapshot is based on a real resource, administrators can be confident that the configuration is realistic and functional.

Pinning a snapshot alters some fundamental behaviors with drift management in JBoss ON:

- It removes any snapshots that were created before that snapshot. For example, if an administrator decides to pin Snapshot 7, Snapshot 0 (the initial image) through Snapshot 6 are all deleted, and Snapshot 7 becomes the new Snapshot 0.
- It creates a baseline image that every change is compared against rather than keeping a moving tally of changes.
- It changes the behavior of drift alerts (Section 5.8, “Defining Drift Alerts”) so that alerts are sent continually until the system configuration is back in compliance with the pinned snapshot.
- The definition it is pinned to cannot be deleted until the snapshot is unpinned.
- If a snapshot is pinned to a template, then all of the resource-level definitions attached to that template automatically use the pinned snapshot as their baseline.
- Any new file added after a snapshot is pinned (or any file deleted) is going to be reported as a new file in every subsequent snapshot. This is because the new snapshot is always compared against the baseline snapshot, so the file is always new to the baseline.

  There is some logic to prevent drift from reporting the same change incessantly. If **`file1.txt`** is added, the agent creates snapshot 1. When the agent does its next detection run, it recognizes that **`file1.txt`** is not in the baseline, but as long as the SHA for **`file1.txt`** has not changed, the agent does not report it as new drift and does not take a new snapshot. If **`file1.txt`** is modified, however, the agent notices the new SHA and sends a new snapshot — with the modified **`file1.txt`** still listed as a new file, because it is compared against the baseline, not the previous version.

### 5.6.2. When to Pin to a Resource and When to Pin to a Template

When a snapshot perfectly matches the configuration that an administrator desires, it can be associated with a drift definition. That snapshot can be pinned to a resource-level definition or a definition template, and there are slightly different reasons to do one or the other.

- Pinning a snapshot to a resource-level definition establishes a baseline for that resource alone. This makes sense while you are still developing an ideal baseline image or for unique environments that may not transition over to other resources.

  Pinning to a resource definition allows a lot of flexibility. It is easy to pin and unpin and select a new snapshot as the baseline, to let administrators develop an ideal configuration with a minimal impact on drift events, alerting, and monitoring because the changes are contained.
- Pinning a snapshot to a template means that baseline can be applied to every resource that uses that template; it allows that one single snapshot to be used across multiple resources. This is makes sense for any kind of repeatable configuration areas and for production or critical systems which must have consistent configuration.

  Pinning to a template is very powerful for maintaining consistency across an entire infrastructure once an ideal configuration has been developed.

Pinning always takes a snapshot that was created on a specific resource and then promotes it to be the baseline for that definition. So the question is — why does a resource-level snapshot need to be pinned to a template? Why can't a template create and use its own snapshot?

The key is to remember that a drift definition template is associated with a resource *type*. The template is not defined as part of a specific resource.

For a resource-level drift definition, the very first drift detection run creates an initial snapshot based on real and existing files. That initial snapshot can be automatically applied as the baseline, pinned snapshot or any snapshot after the initial can be used as the baseline.

However, a drift definition template (Section 5.3.1, “About Resources and Drift Definition Templates”) is not associated with a resource. Therefore, templates do not have a real set of files to work with and it never has its own snapshots to use. The only way that a drift template can be associated with a snapshot is if a resource-level snapshot is pinned to the template.

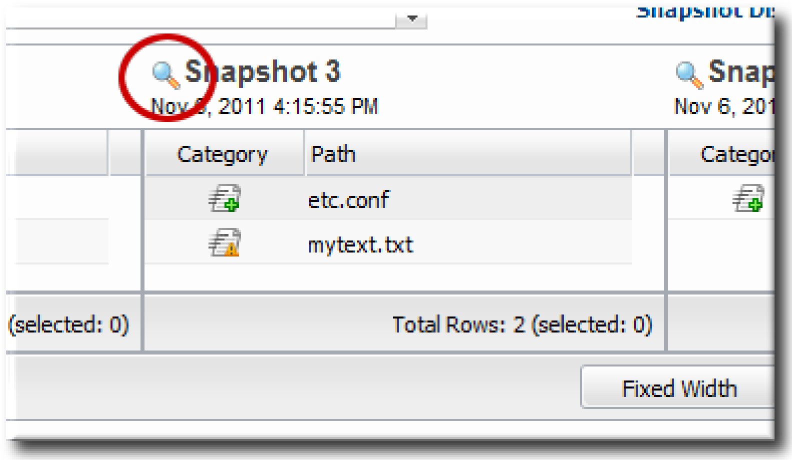
In a sense, pinning a snapshot has a backward workflow from defining a drift definition. A definition starts with a template, then moves to a resource-level definition, which generates a snapshot of that resource. Pinning always begins with a snapshot on a resource, and then moves up to a definition or a definition template.

**NOTE**

A drift definition sets a very clear and limited set of criteria to use for drift detection. When a snapshot is associated with a drift definition template, the template must use the same settings as the original resource-level drift definition which generated the snapshot. If a matching template does not exist, then a new template can be created, using those criteria.

### 5.6.3. Pinning to a Resource-Level Definition

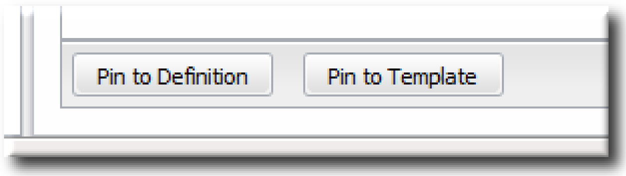
1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab.
4. Click the name of the drift definition.
5. In the snapshot carousel, click the magnifying glass by the name of the snapshot to pin.



**NOTE**

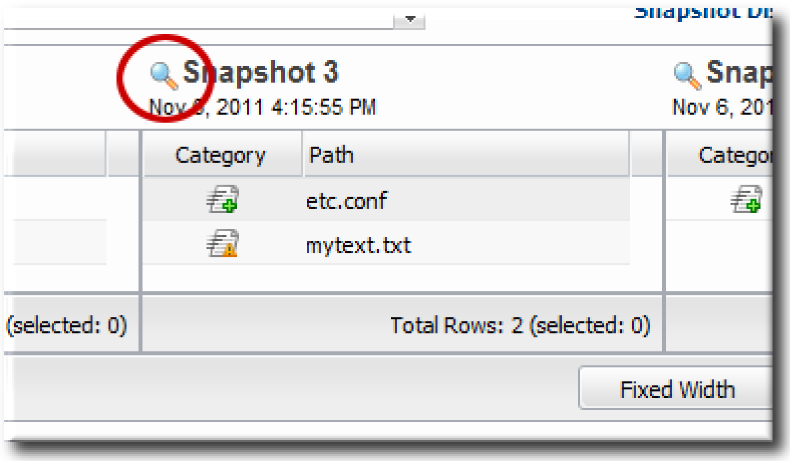
The initial snapshot is not displayed in the carousel. To pin the initial snapshot, click the thumbtack icon in the **`Pinned`** column of the drift definition list. That opens the initial snapshot.
If a snapshot has already been pinned, then clicking the thumbtack icon opens the pinned snapshot.

6. At the bottom of the change list, click the **`Pin to Definition`** button.



### 5.6.4. Pinning to a Template

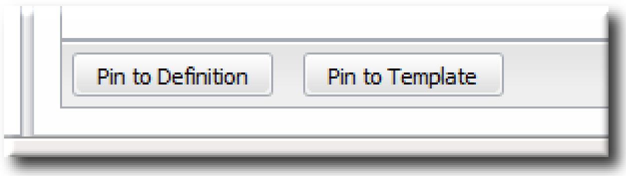
1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab.
4. Click the name of the drift definition.
5. In the snapshot carousel, click the magnifying glass by the name of the snapshot to pin.



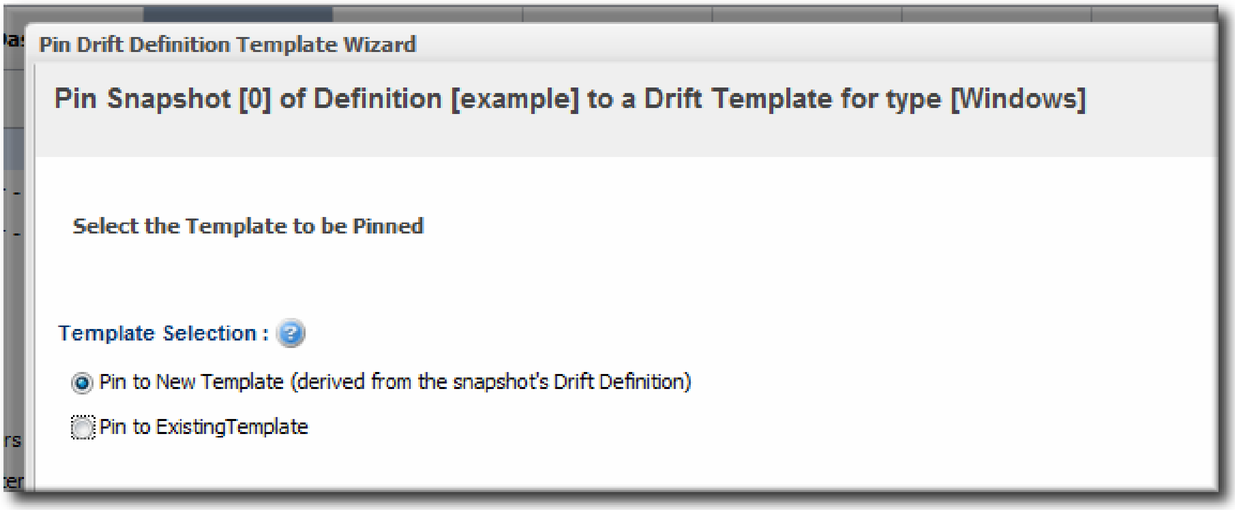
NOTE

The initial snapshot is not displayed in the carousel. To pin the initial snapshot, click the thumbtack icon in the **`Pinned`** column of the drift definition list. That opens the initial snapshot.
If a snapshot has already been pinned, then clicking the thumbtack icon opens the pinned snapshot.

6. At the bottom of the change list, click the **`Pin to Template`** button.



7. If the resource-level template is based on or attached to an existing template, then you can associate the snapshot with that existing template. If the base directory for the resource-level snapshot does not match any existing drift template, then you must create a new template.



Pin Drift Definition Template Wizard

Pin Snapshot [0] of Definition [example] to a Drift Template for type [Windows]

Select the Template to be Pinned

Template Selection :

[✓ Pin to New Template (derived from the snapshot's Drift Definition)]

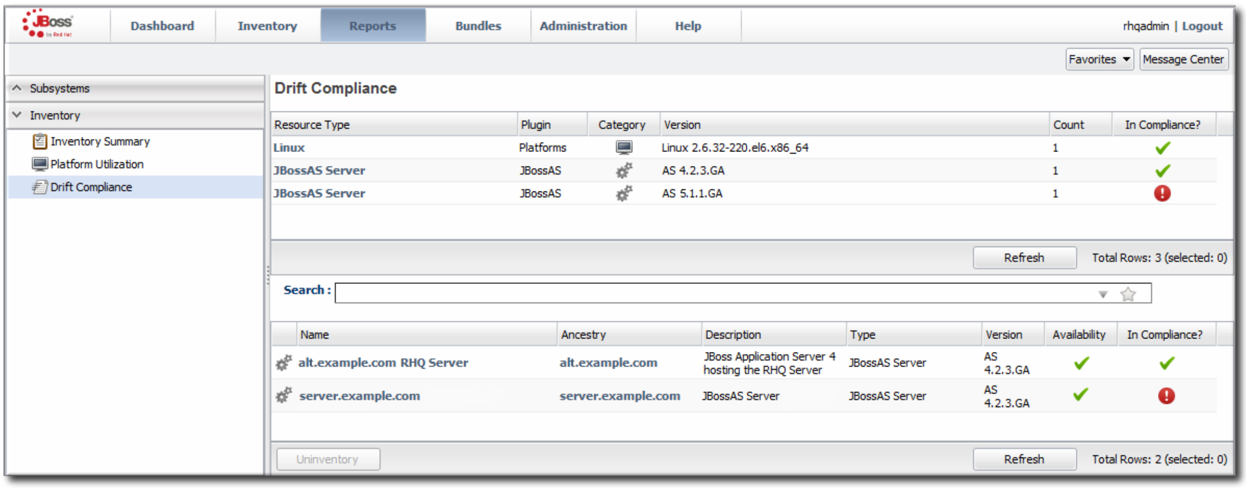
Pin to ExistingTemplate

8. Create the drift template, as in Section 5.3, "Creating a Drift Definition Template".

### 5.6.5. Checking Drift Compliance Reports

The compliance report is a variant of an inventory report. It lists all resources which currently have a drift definition configured and then shows whether they are compliant. Compliance is cumulative; if a resource has multiple drift definitions and is noncompliant on a single one, it will show as non-compliant in the report.

1. Click the **`Reports`** tab in the top navigation menu.
2. Select the **`Drift Compliance`** report from the **`Inventory`** report list.
3. Every resource with a drift definition is listed **by type** and with an icon to indicate whether it is compliant ( ✓ ) or non-compliant ( ❗ ).



Drift Compliance

| Resource Type | Plugin | Category | Version | Count | In Compliance? |
|---|---|---|---|---|---|
| Linux | Platforms | | Linux 2.6.32-220.el6.x86_64 | 1 | ✓ |
| JBossAS Server | JBossAS | | AS 4.2.3.GA | 1 | ✓ |
| JBossAS Server | JBossAS | | AS 5.1.1.GA | 1 | ❗ |

Refresh Total Rows: 3 (selected: 0)

Search :

| Name | Ancestry | Description | Type | Version | Availability | In Compliance? |
|---|---|---|---|---|---|---|
| alt.example.com RHQ Server | alt.example.com | JBoss Application Server 4 hosting the RHQ Server | JBossAS Server | AS 4.2.3.GA | ✓ | ✓ |
| server.example.com | server.example.com | JBossAS Server | JBossAS Server | AS 4.2.3.GA | ✓ | ❗ |

Uninventory Refresh Total Rows: 2 (selected: 0)

4. To get information about the specific resources, click the resource type name; this opens a second inventory report under the main report. All of the resources of that type are listed with their compliance state.

**TIP**

Reports can be exported to CSV, which can be used for office systems or further data manipulation.
To export a report, simply click the **`Export`** button. The report will automatically be downloaded as **`driftCompliance.csv`**.

### 5.6.6. Unpinning a Snapshot

A snapshot can be unpinned — or disassociated — from either a resource-level definition or a drift template. Unpinning a snapshot moves the definition back to a rolling drift detection mode, and any resources that were out of compliance are no longer marked as non-compliant.

1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab.
4. Click the pin icon for the drift definition.

## 5.7. *Extended Example:* Defining Required EAP Configuration

#### The Setup

Tim the IT Guy at Example Corp. has one EAP server running in his production environment. Because of the production load, the EAP server was routinely running out of memory, which was degrading its performance and causing downtime for Example Corp.'s website.

To resolve his immediate memory problem, all Tim has to do is change the heap size setting for his EAP instance. However, Tim needs another strategy for managing the configuration long-term. If he adds another production EAP instance or deploys a new one to replace his current one, it is going to hit the same memory-related performance problems without the new heap size setting.

#### What to Do

There are three things that Tim wants to accomplish to maintain his EAP performance:

- *Find a way to consistently apply configuration to EAP instances.*

  He defines a template for JBoss EAP instances (Section 5.3, “Creating a Drift Definition Template”). To maintain consistency, the template sets the **`Attach to template`** value to true, and each resource-level drift definition will preserve that settings. This ensures that any changes to the template are automatically applied to the JBoss resource drift definitions.
- *Use his current production settings as a basis for future EAP instances.*

  He pins his latest snapshot, with the higher heap settings, to the template definition (Section 5.6.4, “Pinning to a Template”). Every EAP instance is going to be compared against that baseline, so any with the wrong heap setting will immediately be marked out of compliance.
- *Be made aware of specific differences between his current EAP settings and his preferred settings.*

  He creates an alert definition (Section 5.8, “Defining Drift Alerts”) which specifically targets the **`bin/run.conf`** file. This way, he knows precisely whether the heap settings and other JVM settings are wrong for his new instance. He can even use alerts to gather more information about how his EAP instance configuration is different, like using a CLI script to compare the current EAP configuration against the pinned snapshot and then send him the diff.

#### Expected Results

Tim brings a new server online, with a new EAP instance for the production environment. He applies the drift template to the new resource and, within a few minutes, receives a notification that his **`run.conf`** file is not compliant with his preferred configuration. He changes the heap settings on the new EAP instance without having to wait for performance degradation to remember the change.

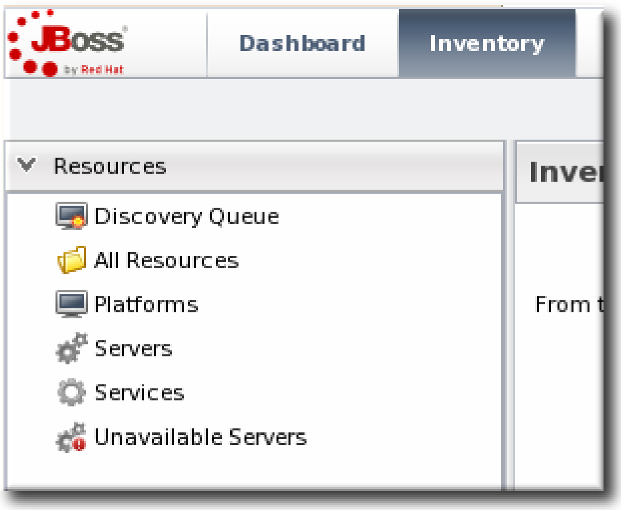
## 5.8. Defining Drift Alerts

Drift changes have their own alert condition.

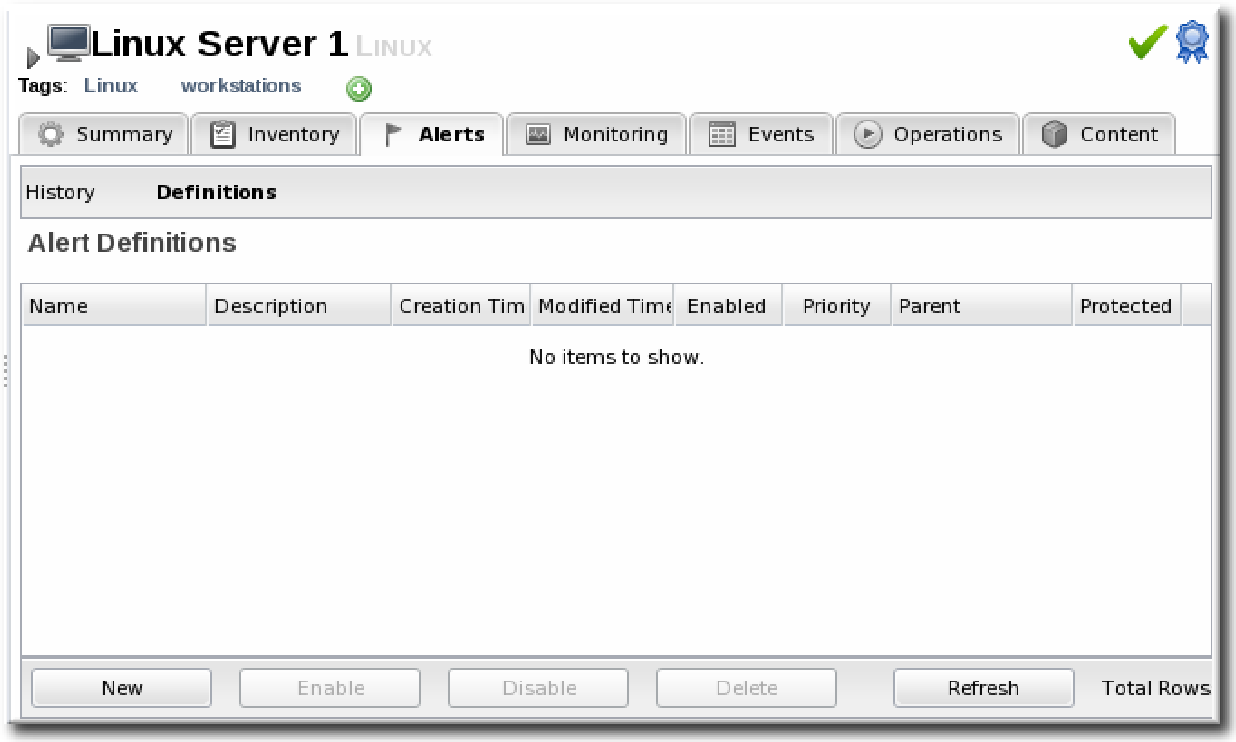
**NOTE**

Recovery alerts are not supported for drift.

1. Click the **Inventory** tab in the top menu.
2. Select the resource type in the **Resources** menu table on the left, and then browse or search for the resource.



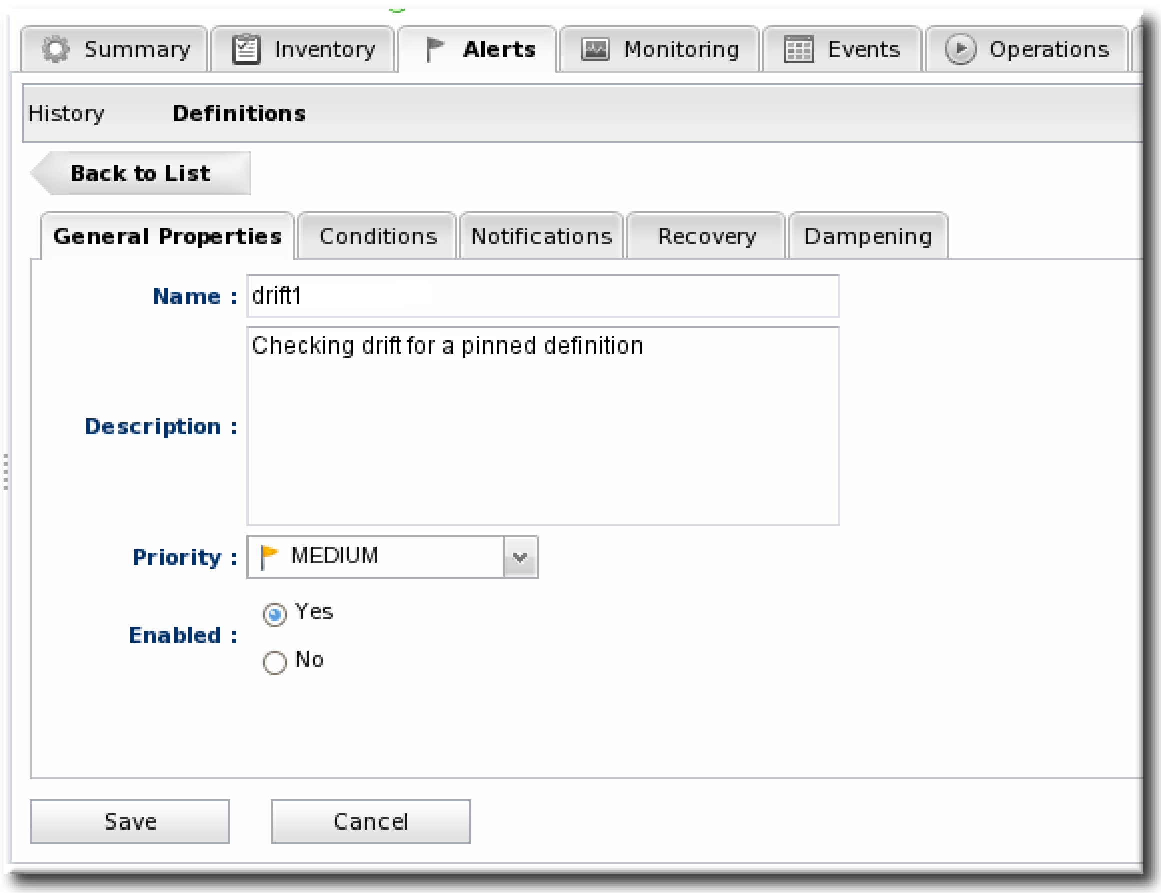
3. Click the resource name in the list.
4. Click the **Alerts** tab for the resource.



| Name | Description | Creation Tim | Modified Time | Enabled | Priority | Parent | Protected |
|---|---|---|---|---|---|---|---|

No items to show.

5. In the **Definitions** subtab, click the **New** button to create the new alert.
6. In the **General Properties** tab, give the basic information about the alert.



Name : drift1

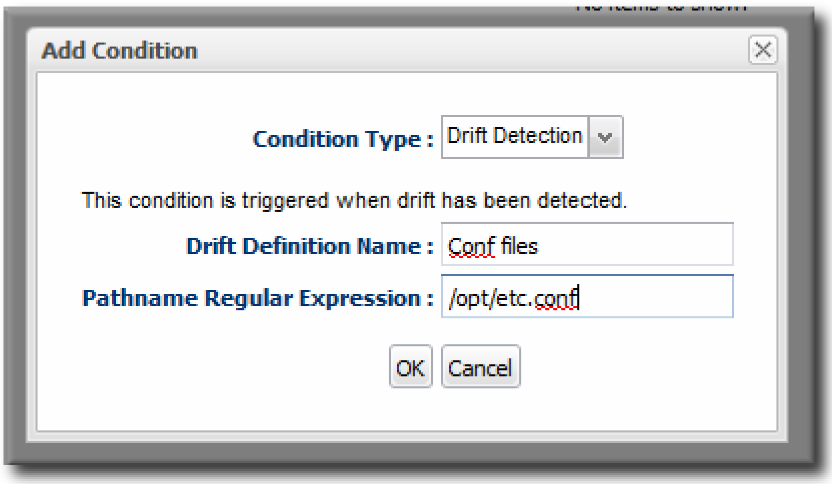
Description : Checking drift for a pinned definition

Priority : MEDIUM

Enabled : [✓ Yes] No

It may be useful to set a *Priority* if the drift definition contains critical configuration files.

7. In the **`Conditions`** tab, select the **`Drift Detection`** option from the conditions list. To use the alert for all drift changes, leave the fields blank. Otherwise, enter the specific drift definition name and (optionally) the directories or files that must be modified for the alert to be triggered.



Condition Type : Drift Detection

This condition is triggered when drift has been detected.

Drift Definition Name : Conf files

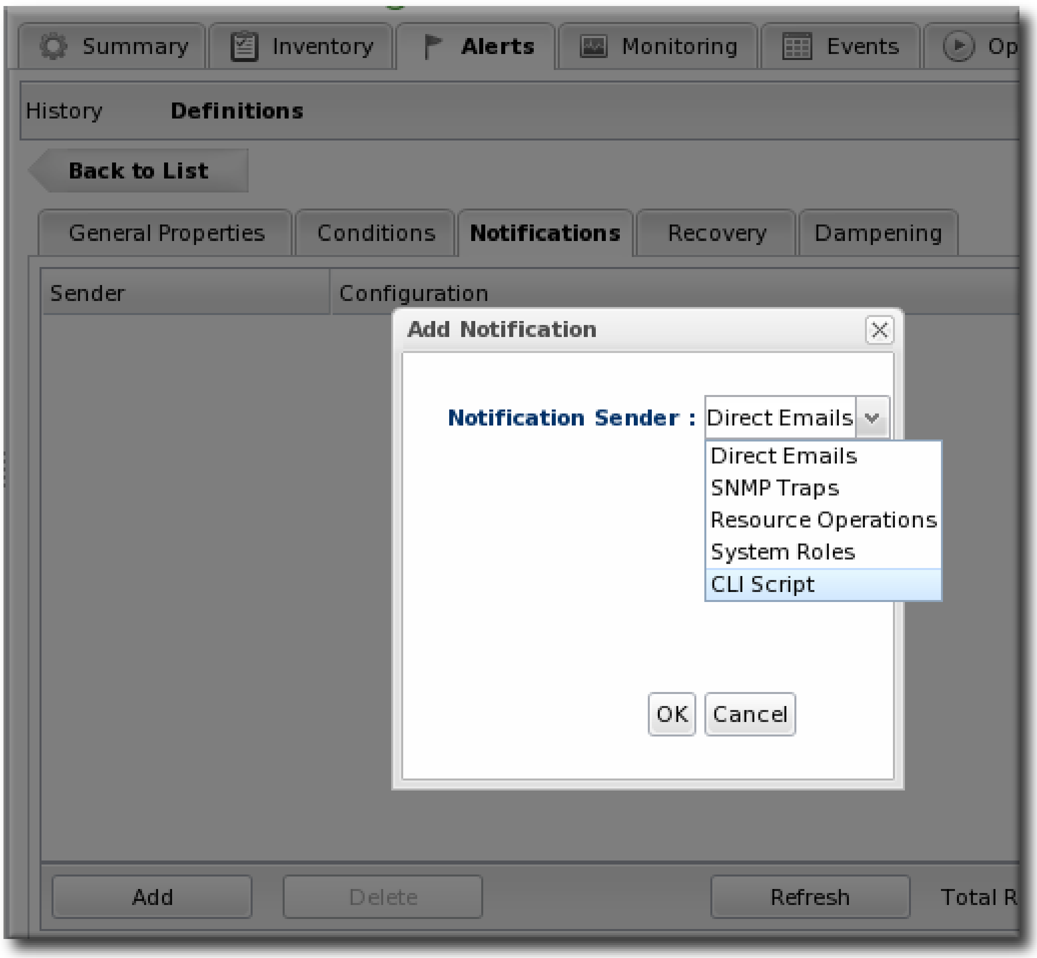
Pathname Regular Expression : /opt/etc.conf

**TIP**

There can be more than one condition set to trigger an alert, meaning that you can use the same alert for multiple drift definitions or files.

8. In the **`Notifications`** tab, click **Add** to set a notification for the alert.

   Select the method to use to send the alert notification in the *Sender* option, and fill in the required information.

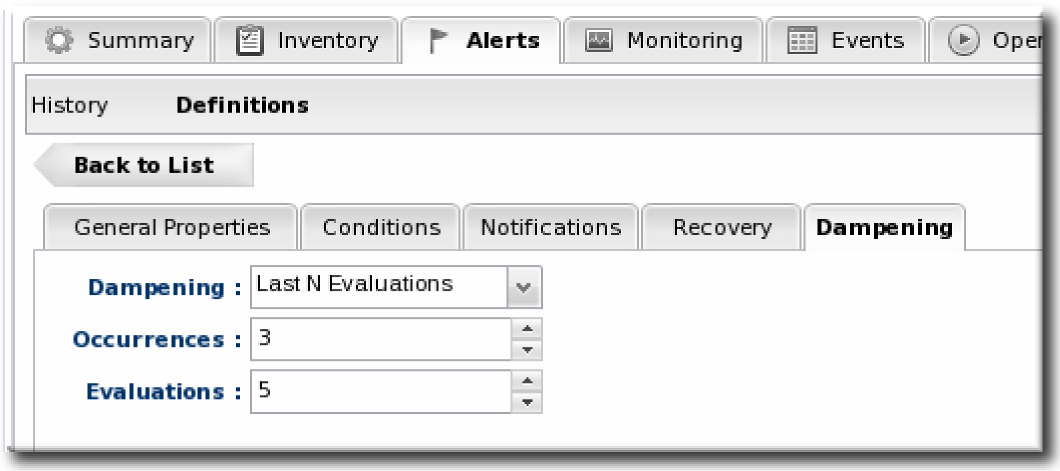


The *Sender* option first sets the specific type of alert method (such as email or SNMP) and then opens the appropriate form to fill in the details for that specific method.

9. Optionally, in the **`Dampening`** tab, give the dampening (or frequency) rule on how often to send notifications for drift.

**TIP**

For pinned snapshots, it can be useful to use dampening rules to keep from getting a flood of alerts before a drift problem is remedied.
Dampening only makes sense for a definition with a pinned snapshot. A pinned definition will fire alerts with every alert scan (every 10 minutes) for as long as it is out of compliance, even if there are no further changes. A rolling definition only fires an alert once, when drift is detected.



Dampening : Last N Evaluations
Occurrences : 3
Evaluations : 5

Any of the dampening rules can be used. The ultimate goal is to limit the number of times that the same alert is set for a resource that is out of compliance with a pinned definition. For example, *Time period* sets a limit on the number of times in a given time period that an alert is issued if the alert condition occurs. Setting the occurrence to 1 and the time period to 4 hours means that when drift is detected once, the server sends an alert and then waits another 4 hours before sending the next alert.

10. Click **OK** to save the alert definition.

## 5.9. *Extended Example:* Reverting a JBoss Server to Its Original Configuration Using Bundles and Server Scripts

### The Setup

In Section 5.7, "*Extended Example:* Defining Required EAP Configuration", Tim the IT Guy at Example Corp. set up drift templates and alerts to help manage the configuration on his production EAP servers. However, his resolution was done manually. When the drift alert notified him that his EAP server was out of compliance, he edited the **run.conf** directly to adjust the heap size.

Manual updates are fine for small infrastructures or infrequent changes. A better management tool, though, is to automate any remediation required for drift.

### What to Do

The goal is to have JBoss ON respond intelligently to drift without requiring any action from Tim the IT Guy. There are two features that allow automated responses:

- Using *bundles* to provision updated files or applications. A bundle is a ZIP file that contains an Ant recipe and any required content (such as configuration files or JARs) for an application. JBoss ON can provision this content on a platform or a JBoss server in a specified directory.

  More information about provisioning bundles is covered in Deploying Applications and Content.
- Launching JBoss ON CLI scripts in response to an alert. One of the possible alert notifications is a server-side alert sender. A JBoss ON CLI script is loaded as content and stored in the JBoss ON server; when the alert fires, it initiates the specified, stored CLI script.

  More information about writing CLI scripts is covered in Running JBoss ON Command-Line Scripts, and general alert information is covered in Setting up Monitoring, Alerts, and Operations.

There are a few steps to remediation using bundles and CLI scripts:

1. Create a bundle file based on the pinned snapshot configuration. The content of the bundle depends on the needs of the deployment. It can be specific configuration files, like

**`bin/run.conf`**, or it can be a full EAP server.

**TIP**

If the bundle contains the full EAP server, then it can be used to create the initial EAP server.

2. Set up the bundle, the destination, and the compatible group to use with the bundle. (The full procedure is described in Deploying Applications and Content.
3. Deploy the bundle with the full EAP server to create the new EAP instance. (Or, if the bundle only has configuration files, create the EAP instances.)
4. Set up the drift definitions, based on the previously configured template (Section 5.7, "*Extended Example:* Defining Required EAP Configuration"), for the new EAP instance.
5. Create a JBoss ON CLI script (in JavaScript) that will automatically deploy the specified bundle to the appropriate destination. An example is in Example 2, "fix-eap.js Script"; in that script, replace the ***`destinationId`*** and ***`bundleVersionId`*** with the real ID numbers for the destination entry and bundle version entry in JBoss ON.
6. Create an alert definition that triggers on the drift detection condition and uses the CLI script notification type, pointing to the JavaScript file that you created.

### Expected Results

Any time drift is detected on the EAP server, it triggers an alert, same as in Section 5.7, "*Extended Example:* Defining Required EAP Configuration". This time, the alert launches the CLI script in response and automatically deploys the bundle — which already has the approved EAP configuration — to the resource. This means that the EAP server is never more than a few minutes out of compliance, roughly the length of one alert scan. All without requiring intervention from Tim the IT Guy.

**Example 2. fix-eap.js Script**

```
/**
 * If obj is a JS array or a java.util.Collection, each element is passed to
 * the callback function. If obj is a java.util.Map, each map entry is passed
 * to the callback function as a key/value pair. If obj is none of the
 * aforementioned types, it is treated as a generic object and each of its
 * properties is passed to the callback function as a name/value pair.
 */
function foreach(obj, fn) {
  if (obj instanceof Array) {
    for (i in obj) {
      fn(obj[i]);
    }
  }
  else if (obj instanceof java.util.Collection) {
    var iterator = obj.iterator();
    while (iterator.hasNext()) {
      fn(iterator.next());
    }
  }
  else if (obj instanceof java.util.Map) {
    var iterator = obj.entrySet().iterator()
    while (iterator.hasNext()) {
      var entry = iterator.next();
      fn(entry.key, entry.value);
    }
  }
  else {   // assume we have a generic object
    for (i in obj) {
      fn(i, obj[i]);
    }
  }
}

/**
 * Iterates over obj similar to foreach. fn should be a predicate that evaluates
 * to true or false. The first match that is found is returned.
 */
function find(obj, fn) {
  if (obj instanceof Array) {
    for (i in obj) {
      if (fn(obj[i])) {
        return obj[i]
      }
    }
  }
  else if (obj instanceof java.util.Collection) {
    var iterator = obj.iterator();
    while (iterator.hasNext()) {
      var next = iterator.next();
      if (fn(next)) {
        return next;
      }
    }
  }
  else if (obj instanceof java.util.Map) {
    var iterator = obj.entrySet().iterator();
    while (iterator.hasNext()) {
      var entry = iterator.next();
      if (fn(entry.key, entry.value)) {
        return {key: entry.key, value: entry.value};
      }
    }
  }
  else {
    for (i in obj) {
      if (fn(i, obj[i])) {
        return {key: i, value: obj[i]};
      }
    }
  }
  return null;
}

/**
 * Iterates over obj similar to foreach. fn should be a predicate that evaluates
 * to true or false. All of the matches are returned in a java.util.List.
 */
function findAll(obj, fn) {
  var matches = java.util.ArrayList();
  if ((obj instanceof Array) || (obj instanceof java.util.Collection)) {
    foreach(obj, function(element) {
      if (fn(element)) {
        matches.add(element);
      }
    });
  }
  else {
    foreach(obj, function(key, value) {
      if (fn(theKey, theValue)) {
        matches.add({key: theKey, value: theValue});
      }
    });
  }
  return matches;
}

/**
 * A convenience function to convert javascript hashes into RHQ's configuration
 * objects.
 * <p>
 * The conversion of individual keys in the hash follows these rules:
 * <ol>
 * <li> if a value of a key is a javascript array, it is interpreted as
PropertyList
 * <li> if a value is a hash, it is interpreted as a PropertyMap
 * <li> otherwise it is interpreted as a PropertySimple
 * <li> a null or undefined value is ignored
 * </ol>
 * <p>
 * Note that the conversion isn't perfect, because the hash does not contain
enough
 * information to restore the names of the list members.
 * <p>
 * Example: <br/>
 * <pre><code>
 * {
 *   simple : "value",
 *   list : [ "value1", "value2"],
 *   listOfMaps : [ { k1 : "value", k2 : "value" }, { k1 : "value2", k2 :
"value2" } ]
```

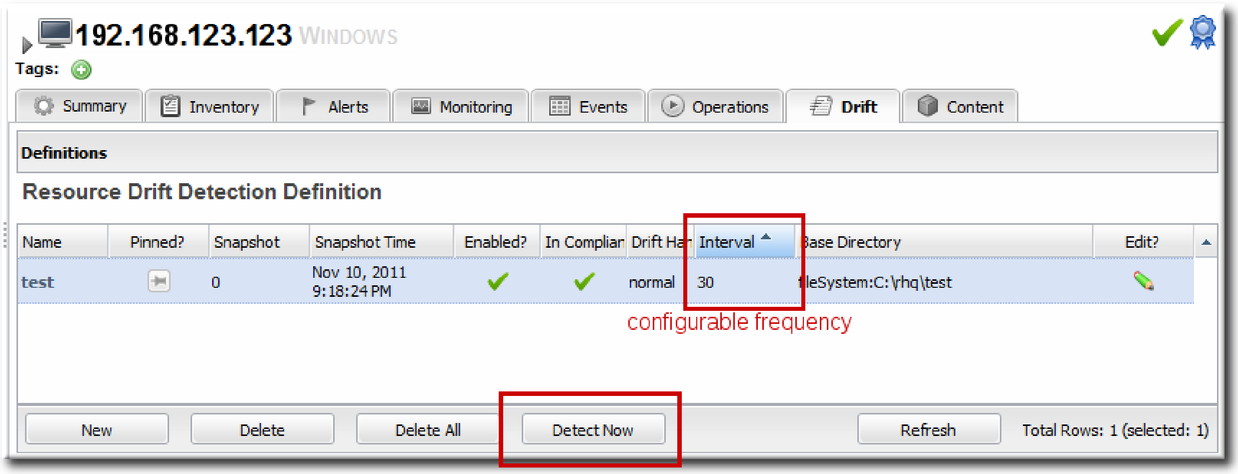
```
 * }
 * </code></pre>
 * gets converted to a configuration object:
 * Configuration:
 * <ul>
 * <li> PropertySimple(name = "simple", value = "value")
 * <li> PropertyList(name = "list")
 *      <ol>
 *      <li>PropertySimple(name = "list", value = "value1")
 *      <li>PropertySimple(name = "list", value = "value2")
 *      </ol>
 * <li> PropertyList(name = "listOfMaps")
 *      <ol>
 *      <li> PropertyMap(name = "listOfMaps")
 *           <ul>
 *           <li>PropertySimple(name = "k1", value = "value")
 *           <li>PropertySimple(name = "k2", value = "value")
 *           </ul>
 *      <li> PropertyMap(name = "listOfMaps")
 *           <ul>
 *           <li>PropertySimple(name = "k1", value = "value2")
 *           <li>PropertySimple(name = "k2", value = "value2")
 *           </ul>
 *      </ol>
 * </ul>
 * Notice that the members of the list have the same name as the list itself
 * which generally is not the case.
 */
function asConfiguration(hash) {

 config = new Configuration;

 for(key in hash) {
  value = hash[key];

  if (value == null) {
   continue;
  }

  (function(parent, key, value) {
   function isArray(obj) {
    return typeof(obj) == 'object' && (obj instanceof Array);
   }

   function isHash(obj) {
    return typeof(obj) == 'object' && !(obj instanceof Array);
   }

   function isPrimitive(obj) {
    return typeof(obj) != 'object';
   }

   //this is an anonymous function, so the only way it can call itself
   //is by getting its reference via argument.callee. Let's just assign
   //a shorter name for it.
   var me = arguments.callee;

   var prop = null;

   if (isPrimitive(value)) {
      prop = new PropertySimple(key, new java.lang.String(value));
     } else if (isArray(value)) {
      prop = new PropertyList(key);
      for(var i = 0; i < value.length; ++i) {
       var v = value[i];
       if (v != null) {
        me(prop, key, v);
       }
      }
     } else if (isHash(value)) {
      prop = new PropertyMap(key);
      for(var i in value) {
       var v = value[i];
       if (value != null) {
        me(prop, i, v);
       }
      }
     }

     if (parent instanceof PropertyList) {
      parent.add(prop);
     } else {
      parent.put(prop);
     }
    })(config, key, value);
   }

   return config;
  }

  /**
   * Opposite of <code>asConfiguration</code>. Converts an RHQ's configuration
  object
   * into a javascript hash.
   *
   * @param configuration
   */
  function asHash(configuration) {
   ret = {}

   iterator = configuration.getMap().values().iterator();
   while(iterator.hasNext()) {
    prop = iterator.next();

    (function(parent, prop) {
     function isArray(obj) {
      return typeof(obj) == 'object' && (obj instanceof Array);
     }

     function isHash(obj) {
      return typeof(obj) == 'object' && !(obj instanceof Array);
     }

     var me = arguments.callee;

     var representation = null;

     if (prop instanceof PropertySimple) {
      representation = prop.stringValue;
     } else if (prop instanceof PropertyList) {
```

```
     representation = [];

     for(var i = 0; i < prop.list.size(); ++i) {
      var child = prop.list.get(i);
      me(representation, child);
     }
    } else if (prop instanceof PropertyMap) {
     representation = {};

     var childIterator = prop.getMap().values().iterator();
     while(childIterator.hasNext()) {
      var child = childIterator.next();

      me(representation, child);
     }
    }

    if (isArray(parent)) {
     parent.push(representation);
    } else if (isHash(parent)) {
     parent[prop.name] = representation;
    }
   })(ret, prop);
  }
  (function(parent) {

  })(configuration);

  return ret;
 }

 /**
  * A simple function to create a new bundle version from a zip file containing
  * the bundle.
  *
  * @param pathToBundleZipFile the path to the bundle on the local file system
  *
  * @return an instance of BundleVersion class describing what's been created on
  * the RHQ server.
  */
 function createBundleVersion(pathToBundleZipFile) {
  var bytes = getFileBytes(pathToBundleZipFile)
  return BundleManager.createBundleVersionViaByteArray(bytes)
 }

 /**
  * This is a helper function that one can use to find out what base directories
  * given resource type defines.
  * <p>
  * These base directories then can be used when specifying bundle destinations.
  *
  * @param resourceTypeId
  * @returns a java.util.Set of ResourceTypeBundleConfiguration objects
  */
 function getAllBaseDirectories(resourceTypeId) {
  var crit = new ResourceTypeCriteria;
  crit.addFilterId(resourceTypeId);
  crit.fetchBundleConfiguration(true);

  var types = ResourceTypeManager.findResourceTypesByCriteria(crit);
 if (types.size() == 0) {
  throw "Could not find a resource type with id " + resourceTypeId;
 } else if (types.size() > 1) {
  throw "More than one resource type found with id " + resourceTypeId + "! How
did that happen!";
 }

 var type = types.get(0);

 return
type.getResourceTypeBundleConfiguration().getBundleDestinationBaseDirectories();
}

/**
 * Creates a new destination for given bundle. Once a destination exists,
 * actual bundle versions can be deployed to it.
 * <p>
 * Note that this only differs from the
<code>BundleManager.createBundleDestination</code>
 * method in the fact that one can provide bundle and resource group names
instead of their
 * ids.
 *
 * @param destinationName the name of the destination to be created
 * @param description the description for the destination
 * @param bundleName the name of the bundle to create the destination for
 * @param groupName name of a group of resources that the destination will
handle
 * @param baseDirName the name of the basedir definition that represents where
inside the
 *                     deployment of the individual resources the bundle will
get deployed
 * @param deployDir the specific sub directory of the base dir where the
bundles will get deployed
 *
 * @return BundleDestination object
 */
function createBundleDestination(destinationName, description, bundleName,
groupName, baseDirName, deployDir) {
 var groupCrit = new ResourceGroupCriteria;
 groupCrit.addFilterName(groupName);
 var groups = ResourceGroupManager.findResourceGroupsByCriteria(groupCrit);

 if (groups.empty) {
  throw "No group called '" + groupName + "' found.";
 }

 var group = groups.get(0);

 var bundleCrit = new BundleCriteria;
 bundleCrit.addFilterName(bundleName);
 var bundles = BundleManager.findBundlesByCriteria(bundleCrit);

 if (bundles.empty) {
  throw "No bundle called '" + bundleName + "' found.";
 }

 var bundle = bundles.get(0);
 return BundleManager.createBundleDestination(bundle.id, destinationName,
description, baseDirName, deployDir, group.id);
}

/**
 * Tries to deploy given bundle version to provided destination using given
configuration.
 * <p>
 * This method blocks while waiting for the deployment to complete or fail.
 *
 * @param destination the bundle destination (or id thereof)
 * @param bundleVersion the bundle version to deploy (or id thereof)
 * @param deploymentConfiguration the deployment configuration. This can be an
ordinary
 * javascript object (hash) or an instance of RHQ's Configuration. If it is the
former,
 * it is converted to a Configuration instance using the
<code>asConfiguration</code>
 * function from <code>util.js</code>. Please consult the documentation of that
method
 * to understand the limitations of that approach.
 * @param description the deployment description
 * @param isCleanDeployment if true, perform a wipe of the deploy directory
prior to the deployment; if false,
 * perform as an upgrade to the existing deployment, if any
 *
 * @return the BundleDeployment instance describing the deployment
 */
function deployBundle(destination, bundleVersion, deploymentConfiguration,
description, isCleanDeployment) {
 var destinationId = destination;
 if (typeof(destination) == 'object') {
  destinationId = destination.id;
 }

 var bundleVersionId = bundleVersion;
 if (typeof(bundleVersion) == 'object') {
  bundleVersionId = bundleVersion.id;
 }

 var deploymentConfig = deploymentConfiguration;
 if (!(deploymentConfiguration instanceof Configuration)) {
  deploymentConfig = asConfiguration(deploymentConfiguration);
 }

 var deployment = BundleManager.createBundleDeployment(bundleVersionId,
destinationId, description, deploymentConfig);

 deployment = BundleManager.scheduleBundleDeployment(deployment.id,
isCleanDeployment);

 var crit = new BundleDeploymentCriteria;
 crit.addFilterId(deployment.id);

 while (deployment.status == BundleDeploymentStatus.PENDING || deployment.status
== BundleDeploymentStatus.IN_PROGRESS) {
  java.lang.Thread.currentThread().sleep(1000);
  var dps = BundleManager.findBundleDeploymentsByCriteria(crit);
  if (dps.empty) {
   throw "The deployment disappeared while we were waiting for it to
complete.";
   }

   deployment = dps.get(0);
  }

 return deployment;
}


var destinationId = 10002;
var bundleVersionId = 10002;
var deploymentConfig = null;
var description = "redeploy due to drift";
// NOTE: It's essential that isCleanDeployment=true, otherwise files that have
drifted will not be replaced with their
//       original versions from the bundle.
var isCleanDeployment = true;
deployBundle(10002, 10002, deploymentConfig, description, true);
```

## 5.10. Running Drift Detection Manually

The drift detection scan runs periodically, according to the interval set in the definition. (The default is 1800 seconds, or 30 minutes.) There can be times when you know that files in the directory have changed and you need a snapshot to be created immediately, but you do not want to change the interval permanently. Simply run a detection scan manually.

1. Click the **`Inventory`** tab in the top menu.
2. Search for the resource.
3. Click the **`Drift`** tab.
4. Select the drift definition to run the scan for.
5. Click the **`Detect Now`** button.



| Name | Pinned? | Snapshot | Snapshot Time | Enabled? | In Compliar | Drift Ha | Interval | Base Directory | Edit? |
|---|---|---|---|---|---|---|---|---|---|
| test |  | 0 | Nov 10, 2011 9:18:24 PM | ✓ | ✓ | normal | 30 | fileSystem:C:\rhq\test |  |

## 5.11. Setting Planned Changes or Disabling Drift Definitions

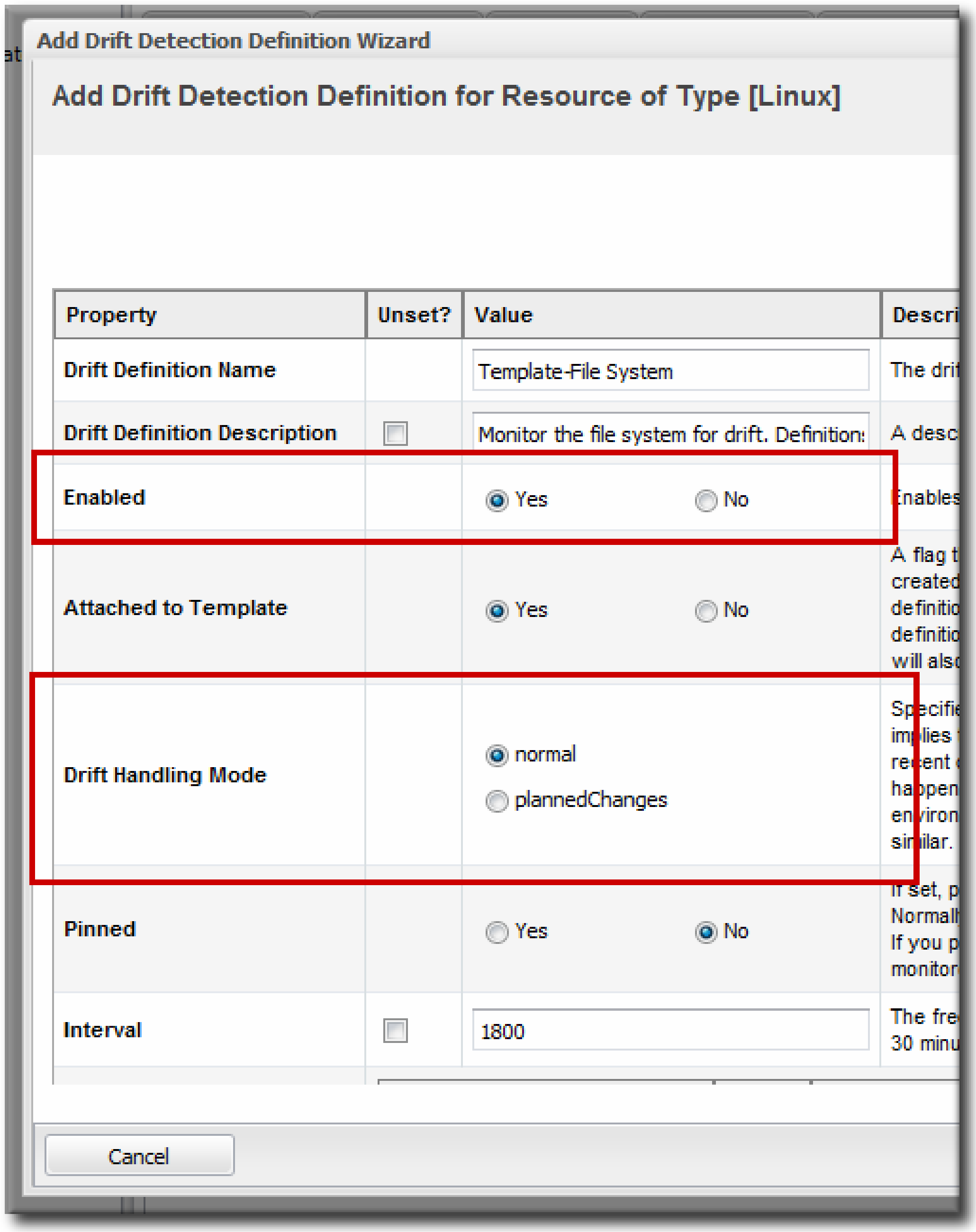
The assumption behind drift monitoring is that there is an identified and specific configuration for a platform or application and that that configuration should be preserved. Changes, therefore, are undesirable and need to be monitored.

However, there can be times when changes are expected, such as scheduled maintenance and upgrade periods. In that situation, it's beneficial to suspend drift monitoring to keep from creating unnecessary static with drift alerts.

There are two ways to suspend drift monitoring:

- Set the drift handling mode to *planned changes*. This keeps running drift detection scans and records changes. Since the changes are expected, though, it doesn't trigger a drift detection event, so it does not issue a drift alert.
- Actually *disable* the drift definition. This suspends the drift detection runs for the definition, not just drift events.

The drift handling mode and the enable option for the drift definition can be edited in the definition entry, as in Section 5.4, "Editing Drift Definitions".



Add Drift Detection Definition Wizard

Add Drift Detection Definition for Resource of Type [Linux]

| Property | Unset? | Value | Descri |
|---|---|---|---|
| Drift Definition Name | | Template-File System | The drif |
| Drift Definition Description | ☐ | Monitor the file system for drift. Definition: | A desc |
| Enabled | | [✓ Yes] No | Enables |
| Attached to Template | | [✓ Yes] No | A flag t created definitio definitio will also |
| Drift Handling Mode | | [✓ normal] plannedChanges | Specifie implies t recent c happen environ similar. |
| Pinned | | Yes [✓ No] | If set, p Normall If you p monitor |
| Interval | ☐ | 1800 | The fre 30 minu |

Figure 8. Drift Handling Mode and Enable Options

## 5.12. Understanding Drift and JBoss ON Agents and Servers

### 5.12.1. Drift Inventory

Both the JBoss ON agent and the JBoss ON server maintain their own inventories of the resources, directories, and files monitored for drift. When the agent starts up, it compares its inventory with the server inventory.

The drift information is stored, with the other agent data, in the ***agentRoot*/rhq-agent/data/** directory. The information in this directory is deleted if the agent is started with new configuration (**--cleanconfig**) or it can be intentionally purged (**--purgedata**). If the drift information is lost, then the agent requests the last snapshot from the JBoss ON server.

The agent always sends the latest changeset to the server as a snapshot. If the server is offline for some period and misses updates, then the agent sends the most current snapshot, which effectively rolls all changes into one snapshot, even if the changes accumulated over several drift detection runs.

### 5.12.2. The Drift Server Plug-in

The server processes drift changes through a server-side plug-in. This plug-in must be enabled for the server to recognize and process drift data sent from the agent.

As with other server-side plug-ins, the drift plug-in can be disabled. However, this effectively and entirely disables drift monitoring on that server, and no drift information is processed or stored. That is slightly different than the behavior of other server subsystems. For example, an individual alert sender can be disabled, but alert detections are still run and alert information is still processed, stored, and displayed by the JBoss ON server.

**WARNING**

If the drift server-side plug-in is disabled, then the server ignores any incoming drift reports. Even if the drift server-side plug-in is re-enabled, any information sent while the plug-in was disabled is lost.

## 5.13. Managing Drift Definitions through the JBoss ON CLI

Drift management tasks can also be performed through the JBoss ON CLI. Examples of some common tasks are in the *Running JBoss ON Command-Line Scripts*, and example scripts are included with the JBoss ON CLI in the *installDir***/rhq-remoting-cli-***version#***/samples/** directory.

The drift API is in the Javadocs at http://docs.redhat.com/docs/en-US/JBoss_Operations_Network/100/html/API_Guides/index.html.

# Index

[1] JBoss ON detects that changes have been made to a binary file. It does not display binary files or compare or diff changes between versions for binary files, only text files.